PUBLIC

SAP HANA Cloud Integration for process integration
2014-11-26

Developer's Guide: Managing Integration Content

Content

1 Managing Integration Content. 4

2 Developing Integration Content Using the Integration Designer. 5

2.1 Configuring the Tool Settings. 5

2.2 Developing Integrations. 6

2.2.1 Creating Integration Project for an Integration Flow. 6

2.2.2 (Optional) Creating a Working Set of Projects. 8

2.2.3 Modifying an Integration Flow Model. 9

2.2.4 Working with Mapping Editor. 10

2.2.5 Importing SAP NetWeaver PI Objects from On-Premise Repository. 16

2.2.6 Configuring an Integration Flow. 18

2.2.7 Defining Exception Subprocesses. 126

2.2.8 Checking the Consistency. 127

2.2.9 Saving Integration Flow as a Template. 128

2.3 Developing Value Mappings. 129

2.3.1 Creating a Value Mapping Project. 129

2.3.2 Editing the Value Mapping Project. 130

2.3.3 Exporting and Importing Value Mapping Groups. 131

2.3.4 Referencing Value Mappings from a Message Mapping. 133

2.3.5 Checking the Value Mapping Consistency. 135

2.4 Externalizing Parameters of Integration Flow. 136

2.4.1 Configuring Multiple Externalized Parameters. 138

2.5 Deploying an Integration Project. 139

2.6 Viewing Error Logs. 140

2.7 Tenant and Integration Flow Tracing. 141

2.8 References to Additional Help. 143

2.9 Understanding the Integration Content Types. 143

3 Packaging Integration Content in SAP HCI Spaces. 146

3.1 Importing Integration Packages. 146

3.2 Creating an Integration Package. 146

3.3 Working with an Integration Package. 148

3.4 Editing an Integration Package. 148

3.5 Locking Integration Packages. 150

3.6 Exporting Integration Packages. 150

4 Accessing Integration Content in SAP HCI Spaces. 152

4.1 Viewing Integration Flow Configurations. 152

4.2 Configuring Multiple Integration Flows. 153

4.3 Editing Integration Flow Configurations. 154

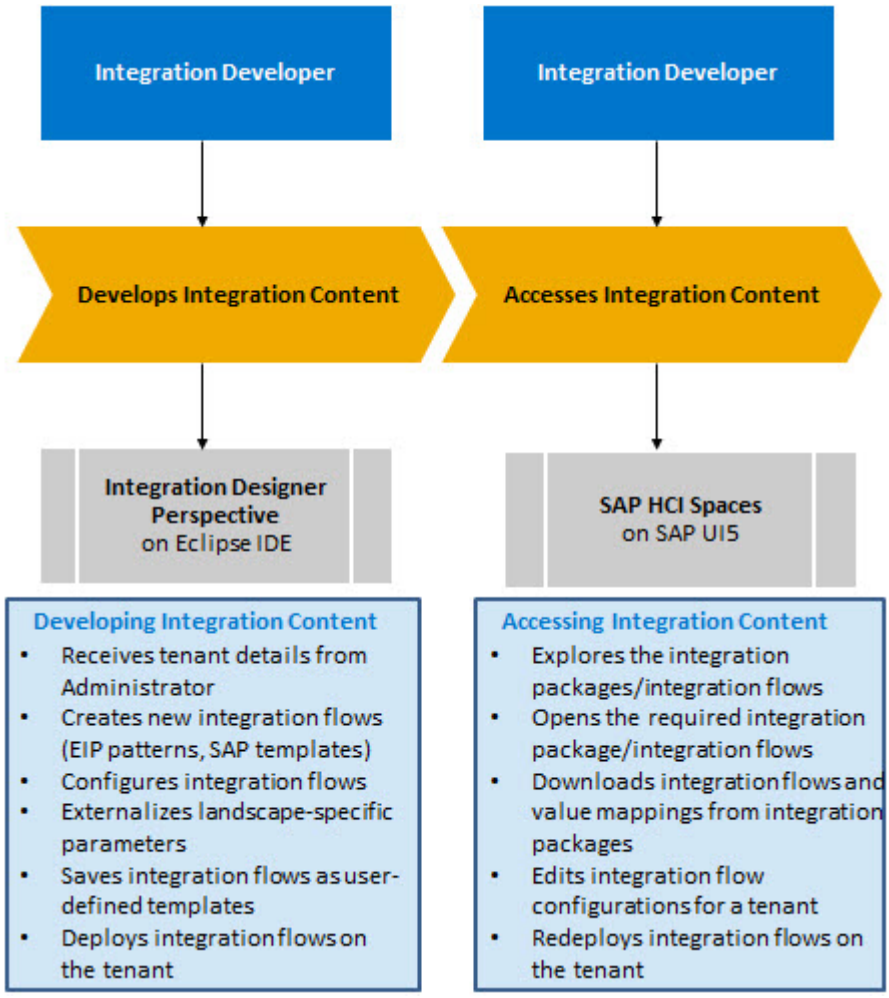
4.3.1 Configuring SuccessFactors Adapter. 155
4.3.2 Configuring SFTP Adapter. 155
4.3.3 Configuring OData Adapter. 156
4.3.4 Scheduling SFSF Adapter for Querying Data from SuccessFactors System. 157
4.3.5 Configuring PGPEncryptor and PGPDecryptor. 158
4.4 Viewing Mapping Details in Mapping Viewer. 160
4.5 Editing Mapping Details. 161
4.6 Changing Source and Target Message Structuring. 163
4.7 Working with Value Mappings. 163
4.8 Deploying Data Flows. 164

1 Managing Integration Content

This topic provides an overview on roles, working environment and tasks involved in managing integration content.

SAP HANA Cloud Integration provides a set of tools and applications that help you perform end-to-end tasks on development and deployment, packaging and publishing, accessing and editing the integration content.

Since the tasks are performed by different roles, in different working environment such as, Integration Designer on Eclipse platform or SAP HCI Spaces on SAP UI5, the figure below helps you understand the relationship between the roles, tools/applications, and tasks:



2 Developing Integration Content Using the Integration Designer

SAP HANA Cloud Integration provides integration tools on the Eclipse platform to model integration flows, configure attributes of the integration flows, and deploy them to the runtime.

You can work with the integration tools in the local development mode, which means that you create an integration project in your local Eclipse workspace and start developing integration content using the features available in the Integration Designer perspective. Once the content is ready, you deploy the project to the runtime in the SAP HANA Cloud Integration infrastructure.

Installing Features of SAP HANA Cloud Integration

To develop and configure integration content, install the features as described on the installation page SAP HANA Cloud Integration Tools.

The Integration Designer feature can be used with the Kepler release (Eclipse 4.3).

2.1 Configuring the Tool Settings

Context

You perform the tasks below to configure the tool settings with attributes you would most likely require when you work with the Eclipse IDE.

Opening the Integration Designer Perspective

1. In the main menu, choose *Windows* > *Open Perspective* > *Other...*.
2. In the *Open Perspective* dialog, select the *Integration Designer*.

Configuring the Operations Server

This setting is required if you are deploying integration flows.

1. Obtain the URL of the Operations Server and the SCN user credentials from SaaS Admin.
2. Enter details at *Window* > *Preferences* > *SAP HANA Cloud Integration* > *Operations Server*.
3. Choose *OK*.

Setting the Connections to the On-Premise Repository

If you need to import interfaces or mappings from an On-Premise repository, such as the ES Repository, you have to set the connection details to establish connection with the ES Repository.

> **Note**
>
> Connection to ES Repository is supported for both Advanced Adapter Engine (AEX) and Dual Stack.

1. In the main menu, choose *Window* > *Preferences*.
2. In the *Preferences* window, select *SAP HANA Cloud Integration* > *Repository Connection*.
3. In the *Repository Connection* page, enter URL of the On-Premise repository in the format `http(s)://<host>:<port>` and enter the user credentials.
4. Choose *Test Connection* to validate the connection.

2.2 Developing Integrations

2.2.1 Creating Integration Project for an Integration Flow

Context

An integration flow is a graphical representation of how the integration content can be configured to enable the flow of messages between two or more participants using SAP HANA Cloud Integration, and thus, ensure successful communication.

You perform this task to create a BPMN 2.0-based integration flow model in your integration project under the **src.main.resources.scenarioflows.integrationflow** package.You can create an integration flow by using the built-in patterns, templates provided by SAP, or user-defined templates.

> **Note**
>
> You can use the templates provided by SAP in the *New Integration Flow* wizard page to help you create and modify integration flows based on certain scenarios. These templates are based on some of the SAP supported scenarios.

> **Restriction**
>
> In this release, SAP supports only limited number of possible integration scenarios.

Procedure

1. In the main menu, choose *Window* > *Open Perspective* > *Integration Designer* to open the perspective.

2. In the main menu, choose *File* > *New* > *Integration Project...* to create a new integration project.
3. In the *New Integration Project* wizard, enter a project name.

> **Note**
>
> By default, *Node Type* is set to IFLMAP, which indicates that the integration flow is deployed to that node in the runtime environment.

4. If you want to add the project to the working set at this point, select the option *Add project to working set*.

> **Note**
>
> If you do not choose to add the project to the working set in this step, you can add it later. For more information about working sets, see Creating a Working Set of Projects [page 8].

5. If you want to create an integration flow of a specific pattern for the new integration project, choose *Next*.

> **Note**
>
> You can also create an integration project together with a point-to-point integration flow. To enable this option, choose *Window* > *Preferences* > *SAP HANA Cloud Integration* > *Integration Flow Preferences* page, and select the *Auto-create integration flow on 'Finish' during integration project creation* option.

6. In the *New Integration Flow* page, enter a name for the integration flow.
7. If you want to create an integration flow using the built-in patterns, select the *Enterprise Integration Patterns* category and choose the required pattern.
8. If you want to create an integration flow using SAP templates, select the *SAP Defined Template* category and choose the required template.
9. If you want to create an integration flow using templates specific to your scenario, select the *User Defined Template* category and choose the required template.

> **Note**
>
> You can find the templates in the *User Defined Template* category only if you have saved an integration flow as a template. For more information, see Saving Integration Flow as a Template [page 128].

10. Choose *Finish*.
    A new project with <integration flow>.iflw is available in the *Project Explorer* view.
11. If you want to provide a description for the integration project, follow the steps below:
    a. In the *Project Explorer*, select the integration project and choose the *Properties* view.
    b. In the *Properties* view, select the *Project Configuration* tab.
    c. In the *Project Configuration* tab page, provide basic details about the integration project, and enter the bundle name and bundle ID.

> **Note**
>
> - The bundle name and bundle ID that you enter get updated in the MANIFEST.MF file.
> - The bundle name and the integration project name are two different attributes.
> - The *Node Type* shows the runtime node type on which the integration flow is deployed.
> - The description field allows you to enter brief information about the integration project to help other users understand the use of the project.

12. Click the graphical area, and in the *Properties* view, select the *Integration Flow* tab page.
13. Enter a description about the integration flow that provides information to other users who will view or work with the integration flow.
14. Save the changes.

2.2.2 (Optional) Creating a Working Set of Projects

Context

You perform this task to group projects using the Working Sets feature provided by Eclipse.

For example, you can create a Working Set for grouping projects based on customer or you can create each Working Set for specific integration scenarios.

> **Note**
>
> The actions available in the context menu of the projects that are added to the Working Set remain as before.

Procedure

1. In the *Project Explorer* view of the *Integration Designer* perspective, select the dropdown icon from the toolbar of the view.
2. Choose *Select Working Set...*
3. In the *Select Working Set* dialog, choose *New...*.
4. In the *New Working Set* dialog, select *Resource* and choose *Next*.
5. Enter a name for the working set.
6. Select a resource from the *Working set contents* section.
7. Choose *Finish*.
8. If you want to edit the working set, select the dropdown icon and choose *Edit Active Working Set*.
9. Select the dropdown icon in the toolbar of the *Project Explorer* and choose *Top Level Elements* > *Working Sets* to display the *Working Set* and its content in the *Project Explorer*.
10. Select *Top Level Elements* > *Projects* to display only the projects belonging to the existing working sets, .

> **Note**
>
> If you want to go back to the default *Project Explorer* view that displays all the projects irrespective of the Working Sets, select the dropdown icon in the toolbar of the *Project Explorer* and choose *Deselect Working Set*.

2.2.3 Modifying an Integration Flow Model

Context

You perform this task if you want to modify the existing integration flow model. For example, if the templates provided by SAP do not exactly match your requirement, you can modify the integration flow created from the templates while adhering to SAP's modeling constraints.

To add integration flow elements, you can drag the notations for systems, connections, mappings, routers, and so on, from the palette and drop it at the required location of the integration flow model in the graphical editor. Alternatively, you can add elements, such as mapping and split, to the integration flow model using the context menu options available on the connections of the model.

> **Note**
>
> The integration flow should match the SAP supported scenarios to avoid deployment failure.

> **Example**
>
> Consider an example which requires you to model an integration flow with multiple pools. The scenario with multiple pool may involve any of the following:
>
> - Hosting same endpoint with different connectors, such as SFTP and SOAP connector
> - Polling content from different servers
> - Grouping similar integration logic that uses different interfaces
>
> The list of elements that you require to model a multiple pool integration flow are:
>
> 1. One *Sender* element
> 2. N *Receiver* elements
> 3. N *Integration Process* pools for each incoming message from the *Sender* to a *Receiver*
> 4. N *Message Flows* from the *Sender* to the *Start Message* element in the *Integration Process* pool. This indicates N incoming message.
> 5. N *Message Flows* from each *End Message* element in the *Integration Process* pool to the corresponding *Receivers*. This indicates N outgoing message flows.
> 6. Finally, *Sequence Flows* to connect the *Start Message* and *End Message* within each pools. This completes the integration flow modeling.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. To modify the graphical diagram using the notations in the *Palette*, follow the substeps below:
   a. If the *Palette* pane is hidden in the *Model Configuration* editor, choose *Show Palette* arrow at the right edge of the editor.

b. Choose the required BPMN notation to modify the integration flow model.

3. To modify the graphical diagram using the context menu options, follow the substeps below:
   a. In the *Model Configuration* editor, right-click on the connections within the pool.
   b. From the context menu, choose the necessary action to add an element. For example, *Add Routing* adds an *Exclusive Gateway* notation to your graphical diagram that can be used either as a receiver router or an interface router.
   c. Save the changes.

2.2.4 Working with Mapping Editor

2.2.4.1 Creating a Message Mapping

Context

You perform this task to create a message mapping when the receiver system accepts a different message format than that of the sender system. The message mapping defines the logic that maps input message structures with the required format of output message structures.

You define a message mapping using the mapping editor by following the steps below:

Procedure

1. Create a mapping object under the package src.main.resources.mapping under the Project Explorer view.
2. Define the signature for message mapping by specifying the source and target elements under the *Signature* section.
3. Define the mapping logic in the *Definition* tab page.

2.2.4.1.1 Multi-Mappings

A multi-mapping is a mapping that allows you to transform a source (input) message to multiple target (output) messages or multiple source to multiple target messages.

Multi-mappings reference multiple message structures. You can, for example, use a multi-mapping to map a message to multiple different (and generally smaller) messages. In that case, the cardinality of the mapping is 1:n.

In your mapping, always add the namespace http://sap.com/xi/XI/SplitAndMerge to the root tag.

For the format of a multi-mapping, see below:

```
<?xml version="1.0" encoding="UTF-8"?>
<sm:Messages xmlns:sm="http://sap.com/xi/XI/SplitAndMerge">
<Message1>
...
</Message1>
<MessageN>
...
</MessageN>
</sm:Messages>
```

2.2.4.1.2 Creating a Mapping Object

Context

You perform this task to create a mapping object required for message mapping.

Procedure

1. In the *Project Explorer*, select the integration project.
2. Expand the integration project and select the src.main.resources. mapping package.
3. From the context menu, choose *New* > *Other...*.
4. In the *New* wizard, select *SAP NetWeaver Cloud Integration* > *Message Mapping*.
5. Choose *Next*.
6. In the *New Message Mapping* wizard, enter a name for the mapping.
7. Choose *Finish*.

Results

The new message mapping object is created under src.main.resources. mapping package and the *Message Mapping Overview* editor opens.

2.2.4.1.3 Defining the Mapping Signature

Context

You perform this task to define the signature of a mapping object.

Procedure

1. In the *Message Mapping Overview* editor, under the *Signature* > *Source Elements* section, *Add* a file.

   > **Note**
   > - If you have selected an XSD or WSDL file, you can also perform multi mapping by selecting another source WSDL or XSD.
   > - If you have selected an EDMX file as your first source element or you are choosing an edmx file as your next source element then you will not be able to perform multi mapping. The new file will overwrite the existing file and only a single file can be seen as source elements.

2. If you are performing multi mapping then change the cardinality of mapping to the required cardinality.
   The default cardinality for single mapping is 1:1.
3. Choose *OK*.
4. Under *Target Elements* section, repeat steps 1 to 3.

2.2.4.1.4 Defining the Mapping

Context

You perform this task to the define a mapping for the mapping object.

Procedure

1. In the *Definition* tab page editor, to map a source field to target field, drag a source node and drop onto a target node in the tree structure.

> **Note**
>
> - Alternately, you right-click on the source field, drag and drop it onto the target node and select *Map automatically*
> - If you have selected multiple messages under source or target elements, system creates a new node *Messages*, which has all the other nodes under it.
>
> .

2. If you want to perform any of the actions such as duplicating the subtree, disabling a field or adding a variable, then from the context menu of the selected node, choose the required option.

   When you need to map different node structures of the source message to only one node structure of the target message you use the option of duplicating the subtrees.
3. If you want to map the child elements of a recursive node (if any), from the context menu choose the option of *Expand Recursive Node*.

> **Note**
>
> - Recursive node acts as a design time placeholder to indicate creation of a node at this location at runtime of the type indicated in the *repeatedNode* property.
> - In case you do not want to use the child elements or variable of the recursive node, then from the context menu, choose the option of *Collapse Recursive Node*.

4. Double-click a node to view the graphical representation of mapping in the graphical editor.

> **Note**
>
> If you want to perform mapping directly in the graphical editor, then drag the source and target nodes from the tree structure into the graphical area.

5. In the graphical editor, connect the source and target node using connectors.
6. If you want to add a function to the connection between the source and target nodes, double-click on the functions displayed under the *Functions* folder on the right-side of the graphical editor.
7. Connect the function to the required node with the use of connectors.
8. If you want to add parameters to a function, choose *Properties* from the context menu of the function.

> **Note**
>
> - You can also double-click on the function to add parameters
> - The functions, in which you can add parameters, are displayed with a wheel symbol

   a. Add a value for the parameter.
   b. Choose *OK*
9. If you want to find the hierarchy of the node in the tree structure, then from the context menu of the node in graphical editor, choose *Find Field*.
10. If you want to change context of the node, then from the context menu, choose *Context* and select an option.
11. If the resulting mapping layout in the graphical editor is not in order, from the toolbar of the *Properties* view, choose  (*Organize Layout*) to automatically arrange the layout.
12. Save the mapping.

> **Note**
>
> To check the correctness of the mapping, you can right-click on the editor and choose *Execute Checks*, if there are any errors, you get notified by a red error mark close to the field or in *Problems* view. Some such errors are described below:

- In case, there are any unassigned mandatory fields, an error is indicated by a red error mark appears close to such fields in the *Definition* tab page.
- If source or target messages are not assigned, an error message is displayed in the *Problems* view.
- If any of the source or target messages are not available in required location, an error mark in red appears on the element in the *Overview* tab page.
- Also, if the mapping is not present in the folder **src.main.resources.mapping** then an error is shown in the *Problems* view.
- The above errors are also displayed once you have saved the mapping.

2.2.4.2 Handling Inconsistencies in Mapping Editor

Context

You use this task to handle the inconsistencies in the message strucutres or mappings that occur due to modification of source or target elements.

Scenarios where such inconsistencies occur are:

- When you change an XSD or WSDL that are being used in existing mapping as source or target element
- When you change the source or target nodes in the message mapping tree structure

> **Note**
>
> Whenever you open a mapping where any changes have been made to the source or target element, the editor notifies you about the change.

Procedure

1. In the *Definition* tab page editor, choose the icon  (*Check for consistencies between changed message structures*).
2. In the *Reload Confirmation* dialog, choose *OK* to continue with correcting of structural inconsistencies.

> **Note**
>
> This action discards any recent changes or additional mappings that you have done after changing the source or target element.

3. In the *Missing Fields* wizard, you can choose to reassign or delete the inconsistent fields.

   > **Note**
   >
   > - A red mark appears on the fields with inconsistencies.
   > - If there are changes in both source and target structures, then in the *Missing Fields* wizard, changes in source structure are displayed first and the follow on screen displays the changes in the target structure.

4. If you want to reassign the missing fields, then follow the substeps below:
   a. Right-click on the missing field in the old structure.
   b. Drag the missing field from the old structure and drop it on the required field in the new structure.
   c. If you want to map all the children fields related to the selected missing field in old structure with the matching fields in the new structure, choose the option *Map Automatically*.
   d. If you want to map only the single selected field, then choose *Create mapping* option.

   > **Note**
   >
   > - Alternatively, you can simply drag the fields from old structure and drop onto the fields in the new structure.
   > - In case you have moved a disabled missing field, then the mapping also moves along with the field but the new field is not marked as disabled.
   > - In case you have moved a missing field with a variable defined under it, then the variable also appears under the new field, and the mapping of the old variable is assigned to the new variable.

5. If you do not require the missing fields in your mapping, then select the missing field and choose the icon (*Delete Field*) to mark for deletion.

   > **Note**
   >
   > If you want to use the field marked for deletion, then select the icon (*Delete Field* icon) again to cancel the delete action.

6. Choose *Next* to check the missing fields of target structure.
7. Repeat the steps above to reassign or delete the missing fields of target structure.
8. Choose *Finish*.
   Now the fields are reassigned and the fields marked for deletion are removed from the structure.
9. Save the mappings.

2.2.4.3 Exporting Mapping Details to Excel

Context

You use this procedure when you want to see the mapping details offline without signing into the Eclipse or NWDS.

Procedure

1. Select the required mapping from the package *main.resources.mapping* .
2. From the context menu of the selected mapping, choose *Export to Excel*.
3. In the *Save As* dialog, select the location for saving the excel containing the mapping details.
4. Select *Save*.

> **Note**
>
> - If the mapping is exported to the selected location without any error, a dialog box with message *Export to excel is completed*, is displayed.
> - If you are performing *Export to Excel* on the mapping for the second time, then it displays a dialog to replace the earlier excel with the new excel.
> - If you trying to export a mapping, for which an excel is already created and in use by another user, then the excel does not open and an error is displayed.
> - If any target node is disabled in the mapping, then the node is greyed out in excel and it displays the value *Disabled* under the under the corresponding cell of column *Type*.
> - Similarly, if a node or field is recursive, it displays the value *...[Recursive]* under the under the corresponding cell of column *Type*.
> - Also, currently only *.xlsx* format of excel is supported.

2.2.5 Importing SAP NetWeaver PI Objects from On-Premise Repository

Context

You perform this task to import interfaces and mappings from an On-Premise repository, such as the ES Repository, into the integration project. In case of mappings, you can import message mappings (.mmap) and operation mappings (.opmap).

> **Restriction**
>
> See the table below to know about the list of unsupported functionalities of the mappings being imported:

Table 1: Unsupported functionalities of imported mappings

Type of Mappings	Limitations
Message Mapping	User Defined Functions with Channel type parameter
	RFC and JDBC lookups used in Mappings
	Parameters declared through *Parameter* section of mapping editor (in ES Repository Builder)

Type of Mappings	Limitations
	Used Imported Archives
	Used Function Libraries
	Containing schema from .xml or .zip files
Operation Mapping	Operations (source or target) with cardinality '0..unbonded' are not supported
	Operation mappings with cardinality 0..1 are not supported
	Operation mapping containing multiple operations are not supported
	Since parameters are not supported in Message Mapping, so operation mappings with binding are also not supported
	Operation Mappings with Java Mappings (Imported Archives, Java programs) and external programs are not supported
	Operation Mappings, with Do Not Resolve XOP Includes, Read Attachments are not supported
	Only synchronous operation mappings with atleast one mapping program present- either request, response or fault, is supported.
	If the WSDL is already present, WSDLs are not overwritten

Procedure

1. In the *Project Explorer*, right-click on an integration project and from the context menu choose *Import PI Object*.
2. In the *Import PI Object* dialog, select *ES Repository* below the object type you want to import.

   For example, if you want to import operation mappings, select *ES Repository* below *Operation Mapping* object type.
3. Choose *Next*.
4. In the *Import Mappings from ES Repository* dialog, select one or more objects and choose *Finish*.

Results

The imported objects are placed under their respective src.main.resources.<object> folder. For example, check the imported mapping under **src.main.resources.mapping** and imported interface under **src.main.resources.wsdl**.

WSDLs/XSDs corresponding to Message Types and Fault Message Types are placed under **src.main.resources.mapping** folder, other interfaces get placed under **src.main.resources.wsdl**.

The imported operation mapping has the following features:

- If operation mapping contains message mapping, then the message mapping is downloaded as a jar under src.main.resources.mapping package.
- If the operation mapping contains XSLTs, then the files are downloaded as .xsl under *src.main.resources.mapping*.
- Imported source or target WSDLs are not supported in integration flows.

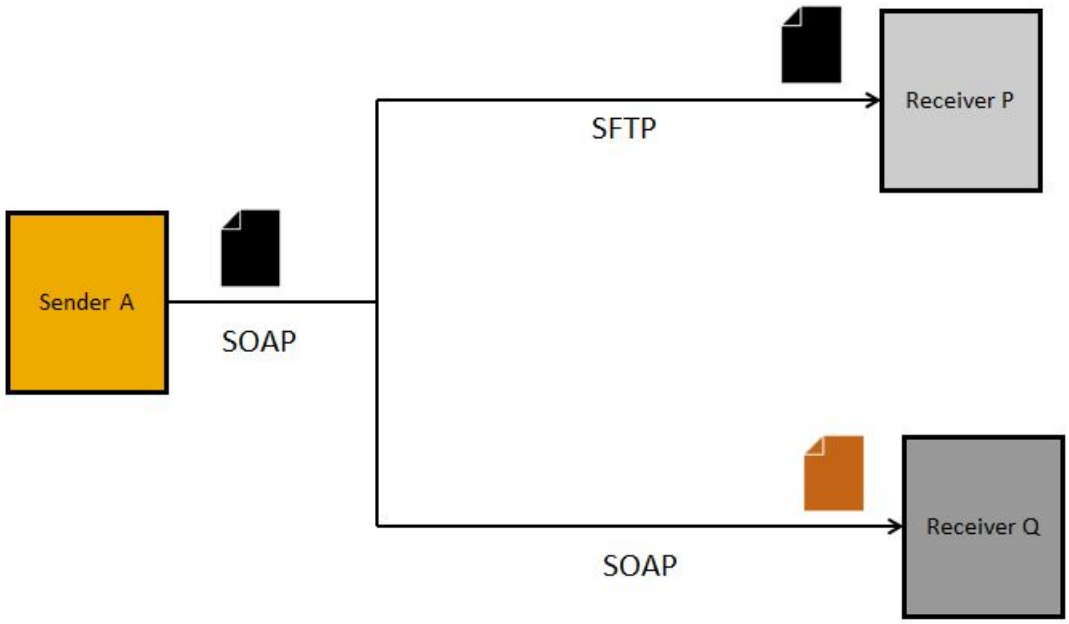
2.2.6 Configuring an Integration Flow

Context

You perform this task to configure an integration flow to represent a specific scenario.

You configure the integration flow by adding elements to the graphical model and assigning values to the elements relevant to the scenario. The basic integration flow requires you to configure the sender and receiver channels to enable a point-to-point process flow from the sender to a receiver.

The figure below helps you understand how a scenario is configured using an integration flow and is followed by an explanation:

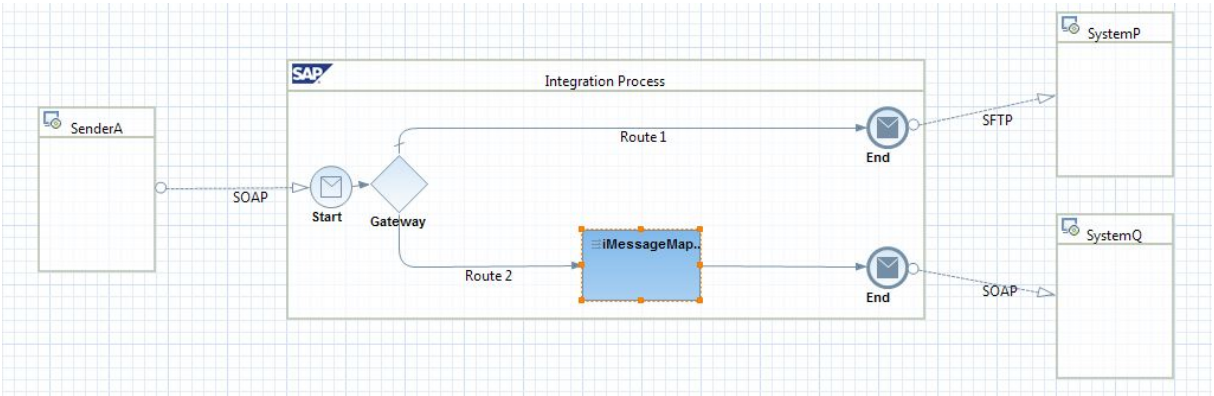


The scenario involves communication of System A with System P and System Q, where System A sends messages to System P and System Q.

System A and System P have different communication protocols, whereas, System Q requires additional field information in the message format sent by System A. In such a case, you do the following configurations in the integration flow:

- Create an integration flow with a gateway branching out to two receivers.
- Configure conditions to route messages to the correct receiver.
- Place a mapping component in the communication between System A and System Q

After configuration, the resulting integration flow should be similar to the example shown below:



2.2.6.1 Assigning the Sender and Receiver Participants

Context

You perform this task to assign the sender participant and receiver participant to the integration flow. To allow the sender participant to connect to the Enterprise Service Bus, you have to provide either the client certificates or authenticate using the SDN username and password.

Procedure

1. Assign the Sender Participant
   a. In the *Model Configuration* editor tab page, select the sender.
   b. In the *Properties* page, enter a name for the sender system that may represent a single participant or a group of logically related participants in a communication.
   c. Either select *Basic Authentication* option or provide a client certificate in the *Sender Authorization* table to authenticate the sender.

      > **Note**
      >
      > You can either browse for the client certificate, for example <UserID>.crt, from your local file system or add and enter the *Subject DN*(information used to authorize the sender) and *Issuer DN*(information about the Certificate Authority who issues the certificate) manually.

   d. In the *Sender Authorization* section, choose *Add...* to browse and add an authorized client certificate or enter the *Subject DN* and *Issuer DN* manually.

      > **Note**
      >
      > To quickly add an authorized client certificate from the *Model Configuration* editor tab page, select *Import Certificate* from the context menu of the sender element.

   e. Save the changes.
2. Assign the Receiver Participant
   a. In the *Model Configuration* editor page, select the receiver.
   b. In the *Properties* page, enter a name for the receiver system.
   c. Save the changes.

2.2.6.2 Defining Channels

Prerequisites

- You have configured connections to an On-Premise repository if you have to obtain interface WSDL from the repository into this project.

  > **Note**
  >
  > You can import service interfaces from ES Repository with server version 7.1 onwards. The imported service interface WSDLs get added to **src.main.resources.wsdl**.

  For more information on setting the connections to the repository, see *Setting the Connections to On-Premise Repository*under Configuring the Tool Settings [page 5].
- If you want to use a WSDL available in your local file system, you have copied the file under **src.main.resources.wsdl** in your project.

Context

You perform this task to enable communication between the sender and receiver participants by defining connectors for the sender and receiver channels of the integration flow.

Procedure

1. In the *Model Configuration* editor page, select the sender or receiver channel (dotted lines at sender and receiver side).
2. To configure the channel with a saved configuration that is available as a template, choose *Load Channel Template* from the context menu of the channel.
3. To specify new configurations, follow the instructions mentioned in the adjacent topic for the required connector.

   > **Tip**
   >
   > If you want to reuse the connector configurations for channels that are within or across integration flows, then select the *Copy* and *Paste* option from the context menu of the channel.
4. To save the configurations of the channel as a template, choose *Save as Template* from the context menu of the channel.

   > **Note**
   >
   > When you save the configuration of the channel as a template:

- The tool stores the template in the workspace as <ElementTemplateName>.fst.
- The tool saves the parameter key of the externalized parameters and not the values.

2.2.6.2.1 Configuring a Channel with IDoc (IDoc SOAP) Adapter

Context

The IDoc with SOAP message protocol is used when a system needs to exchange IDoc messages with another system that accepts data over SOAP protocol.

Procedure

1. If you are configuring the sender channel, ensure the sender authorization certificate is specified by following the steps:
   a. In the *Model Configuration* editor, select the sender.
   b. In the *Properties* view, check if the certificate is available in the *Sender Authorization* table, or add a certificate.
2. In the *Model Configuration* editor, double-click the sender or receiver channel.
3. Choose the *General* tab page and enter the details listed below.
4. In the *Adapter Type* field, browse and select the *IDoc* adapter and *Message Protocol* as *IDoc SOAP*.
5. Choose the *Adapter Specific* tab page and enter the details as shown in the table below:

Table 2: Parameters and Values of Sender IDoc (IDoc SOAP) Adapter

Section	Parameters	Description
Connection Details	*Address*	Relative endpoint address on which ESB Bus listens to the incoming requests, for example, */HCM/GetEmployeeDetails*.

Section	Parameters	Description
	URL to WSDL	The WSDL can be downloaded for a particular integration flow using its bundle ID and the corresponding endpoint as parameters. Here, the WSDL endpoint is the relative path of the hosted endpoint. Note the following: ○ For newly deployed integration flows, the WSDL that is generated by the download corresponds to the endpoint configuration in the integration flow. ○ When in the integration flow the corresponding parameter `includePolicies=true` is set, the downloaded WSDL includes policies. Note that some systems might not be able to consume WSDLs including policies correctly. ○ For WS consumer based on SAP AS ABAP, the parameter abapConsumer is required. Note that this parameter is set to `abapConsumer=false` by default. ○ As an example, a WSDL URL (including parameters) can be: `https://<servername>/Operations/api/WSDLDownload?artifactName=IVY_WSRM_Connectivity&servicePath=/orange/ivy&domainName=bsn.neo.ondemand.com&includePolicies=false&abapConsumer=true`

Table 3: Parameters and Values of Receiver IDoc (IDoc SOAP) Adapter

Section	Parameters	Description
Connection Details	*Address*	Endpoint address on which the ESB Bus posts the outgoing message, for example http://<host>:<port>/payment.
	URL to WSDL	URL to the WSDL defining the WS provider endpoint (of the receiver). You can provide the WSDL by: ○ Directly entering /wsdl/<interfacename>.wsdl where the WSDL is available in src.main.resources.wsdl. ○ Selecting a source to browse for a WSDL, either from an On-Premise ES Repository or your local workspace.

Section	Parameters	Description
	IDoc Content Type	*Application/x-sap.doc* ○ Allows only single IDoc record for each request ○ Enables Exactly-Once processing ○ Enables message sequencing *Text/XML* ○ Allows multiple IDoc records for each request
	Request Timeout	Specifies the time (in milliseconds) that the client will wait for a response before the connection is being interrupted. The default value is 60000 miliseconds (1 minute).
	Compress Message	Enables the WS endpoint to send compressed request messages to the WS Provider and to indicate the WS Provider that it can handle compressed response messages.
	Allow Chunking	Used for enabling HTTP chunking of data while sending messages.
	Connect using Basic Authentication	Select this option to allow the ESB to connect to the receiver system using the deployed basic authentication credentials. When you select this option, you need to provide the credential name. *Credential Name*: Enter the credential name of the username-password pair specified during the deployment of basic authentication credentials on the cluster.

Section	Parameters	Description
	Private Key Alias	Allows you to enter the private key alias name that gets the private key from the keystore and authenticates you to the receiver in an HTTPs communication. If you have selected the option of *Connect using Basic Authentication*, this field is not visible.

6. Save the changes.

Results

In the *Model Configuration* editor, when you place the cursor on the sender or receiver message flows, you can see the *SOAP Address* and *WSDL* information.

2.2.6.2.2 Configuring a Channel with SOAP (SAP RM) Adapter

Context

You perform this task to configure a sender or receiver channel with the SOAP communication protocol, with SAP RM as the message protocol. SAP RM is a simplified communication protocol for asynchronous Web service communication that does not require the use of Web Service Reliable Messaging (WS-RM) standards. It offers a proprietary extension to ensure reliability on the ABAP back-end side of both Web service consumers and providers. For more information, see http://wiki.scn.sap.com/wiki/display/ABAPConn/Plain+SOAP

Procedure

1. Choose the *General* tab page and enter the details below.
2. In the *Adapter Type* field, browse and select the *SOAP* adapter, and *SAP RM* as the *Message Protocol*.
3. Choose the *Adapter Specific* tab page and enter the details as shown in the table below:

Table 4: Parameters and Values of Sender SOAP (SAP RM) Adapter

Section	Parameters	Description
Connection Details	*Address*	Relative endpoint address at which the ESB listens to the incoming requests, for example, */HCM/GetEmployeeDetails*.
	URL to WSDL	The WSDL can be downloaded for a particular integration flow using its bundle ID and the corresponding endpoint as parameters. Here, the WSDL endpoint is the relative path of the hosted endpoint. Note the following: ○ For newly deployed integration flows, the WSDL that is generated by the download corresponds to the endpoint configuration in the integration flow. ○ When in the integration flow the corresponding parameter `includePolicies=true` is set, the downloaded WSDL includes policies. Note that some systems might not be able to consume WSDLs including policies correctly. ○ For WS consumer based on SAP AS ABAP, the parameter abapConsumer is required. Note that this parameter is set to `abapConsumer=false` by default. ○ As an example, a WSDL URL (including parameters) can be: `https://<servername>/Operations/api/WSDLDownload?artifactName=IVY_WSRM_Connectivity&servicePath=/orange/ivy&domainName=bsn.neo.ondemand.com&includePolicies=false&abapConsumer=true`
	Robust One Way Communication	Used for reliable one-way message exchanges. The consumer sends a message to a provider (in this case the channel), which returns a status. A fault is returned if processing fails.

Table 5: Parameters and Values of Receiver SOAP (SAP RM) Adapter

Section	Parameters	Description
Connection Details	*Address*	Endpoint address at which the ESB posts the outgoing message, for example http://<host>:<port>/payment.
	URL to WSDL	URL to the WSDL defining the WS provider endpoint (of the receiver). You can provide the WSDL as follows: ○ Enter /wsdl/<interfacename>.wsdl directly if the WSDL is available in src.main.resources.wsdl. ○ Select a source to browse for a WSDL either from an on-premise ES Repository or your local workspace.
	Service Name	Name of the selected service contained in the referenced WSDL.

Section	Parameters	Description
	Port Name	Name of the selected port of a selected service (that you provide in the Service Name field) contained in the referenced WSDL. **Note** Using the same port names across receivers is not supported. To use the same port names, you need to create a copy of the WSDL and use it.
	Operation Name	Name of the operation of the selected service (that you provide in the Service Name field) contained in the referenced WSDL.
	Private Key Alias	Allows you to enter the private key alias name that gets the private key from the keystore and authenticates you to the receiver in an HTTPS communication. **Note** If you have selected the *Connect using Basic Authentication* option, this field is not visible.
	Compress Message	Enables the WS endpoint to send compressed request messages to the WS provider and to indicate to the WS provider that it can handle compressed response messages.
	Allow Chunking	Used for enabling HTTP chunking of data while sending messages.
	Request Timeout	Specifies the time (in milliseconds) that the client will wait for a response before the connection is interrupted. The default value is 60000 milliseconds (1 minute).
	Connect using Basic Authentication	Select this option to allow the ESB to connect to the receiver system using the deployed basic authentication credentials. If you select this option, you need to provide the credential name. *Credential Name*: Enter the credential name of the user name-password pair specified during the deployment of basic authentication credentials on the cluster.

4. Save the configurations in both the sender and receiver channel editors.

Results

In the *Model Configuration* editor, when you place the cursor on the sender or receiver message flows, you can see the *SOAP Address* and *WSDL* information.

2.2.6.2.3 Configuring a Channel with SOAP (SOAP 1.x) Adapter

Prerequisites

Since the adapter implements web services security, you have ensured that the related certificates are deployed in the truststore.

Context

SOAP (SOAP 1.x) allows you to deploy web services that support SOAP 1.1 and SOAP 1.2. SOAP 1.x provides you a framework for binding SOAP to underlying protocols. The binding specification in the WSDL defines the message format and protocol details for a web service.

Procedure

1. If you are configuring the sender channel, ensure the sender authorization certificate is specified by following the steps below:
   a. In the *Model Configuration* editor, select the sender.
   b. In the *Properties* view, check if the certificate is available in the *Sender Authorization* table, else add a certificate.
2. In the *Model Configuration* editor, double-click the sender or receiver channel.
3. In the *Adapter Type* section of the *General* tab page, select *SOAP* from the *Adapter Type* dropdown and select SOAP 1.x as the message protocol.
4. Choose the *Adapter-Specific* tab page and enter the details as shown in the table below:

Table 6: Parameters and Values of Sender SOAP (SOAP 1.x) Adapter

Section	Parameters	Desciption
Connection Details	*Address*	Relative endpoint address on which ESB Bus listens to the incoming requests, for example, "*/HCM/GetEmployeeDetails*".

Section	Parameters	Desciption
	URL to WSDL	The WSDL can be downloaded for a particular integration flow using its bundle ID and the corresponding endpoint as parameters. Here, the WSDL endpoint is the relative path of the hosted endpoint. Note the following: ○ For newly deployed integration flows, the WSDL that is generated by the download corresponds to the endpoint configuration in the integration flow. ○ When in the integration flow the corresponding parameter `includePolicies=true` is set, the downloaded WSDL includes policies. Note that some systems might not be able to consume WSDLs including policies correctly. ○ For WS consumer based on SAP AS ABAP, the parameter abapConsumer is required. Note that this parameter is set to `abapConsumer=false` by default. ○ As an example, a WSDL URL (including parameters) can be: `https://<servername>/Operations/api/WSDLDownload?artifactName=IVY_WSRM_Connectivity&servicePath=/orange/ivy&domainName=bsn.neo.ondemand.com&includePolicies=false&abapConsumer=true`
	Service Name	Name of the selected service contained in the referenced WSDL
	Port Name	Name of the selected port of a selected service (that you provide in the Service Name field) contained in the referenced WSDL
	Processing Settings	○ Standard: Message is executed with standard processing mechanism ○ Robust: Provider invokes service synchronously and the processing errors are returned to the consumer.
WS-Security(Web Services Security)	*WS-Security Configuration*	Specifies the way how WS-Security settings are to be configured. ○ *Via Manual Configuration in Channel* The security settings are manually to be configured (see below listed attributes). ○ *None* No WS-Security is applied for message exchange. If you select this option, no further WS-Security-relevant settings can be applied.

Section	Parameters	Desciption
	WS-Security Type	Specifies the combination of message protection methods that are to be applied. There are the following options: ○ *Verify Message* A signed payload is expected by the tenant, and the signature has to be verified. ○ *Verify and Decrypt Message* A signed and encrypted payload is expected by the tenant. The signature has to be verified and the payload has to be decrypted.
	Save Incoming Signed Message	Select this option if the incoming signed (and encrypted) message is to be stored.
	Check Time Stamp	Select this option, if the sender (WS consumer) sends a time stamp along with the message. In case a request-response pattern is configured, a time stamp is added to the response message.
	*Sender is Basic Security Profile Compliant*t	As default setting, select this option. This option should be selected in case the sender system supports a dedicated security level which is described by the Basic Security Profile (as assumed to be the case for most systems). Only in case the sender system does not support this security profile, de-select this option.
	Private Key Alias for Response Signing	Specify an alias for the private key that is to be used to sign the response message. The tenant private key is used to sign the response message (that is sent to the WS consumer). The tenant private key has to be part of the tenant keystore.
	Public Key Alias for Response Encryption (only in case **`Verify and Decrypt Message`** is selected for *WS-Security Type*)	Specify an alias for the public key that is to be used to encrypt the response message. The sender (WS consumer) public key is used to sign the response message (that is sent to the WS consumer). This key has to be part of the tenant keystore.
	Initiator Token	○ *Include Strategy* Pre-set value **`Always to Recipient`** ensures that the WS consumer sends the certificate along with the message. ○ *X509 Token Assertion* Defines the format of the certificate being sent by the WS consumer along with the message.

Section	Parameters	Desciption
	Recipient Token	*Include Strategy* Value **`Always to Initiator`** adds the certificate to the response message.
	Algorithm Suite Assertion	Specifies which algorithms are to be used by the WS consumer.

Table 7: Parameters and Values of Receiver SOAP (SOAP 1.x) Adapter

Section	Parameters	Desciption
Connection Details	*Address*	Endpoint address on which the ESB Bus posts the outgoing message, for example http://<host>:<port>/payment.
	URL to WSDL	URL to the WSDL defining the WS provider endpoint (of the receiver). You can provide the WSDL by: ○ Directly entering /wsdl/<interfacename>.wsdl where the WSDL is available in src.main.resources.wsdl. ○ Selecting a source to browse for a WSDL either from an On-Premise ES Repository or your local workspace.
	Service Name	Name of the selected service contained in the referenced WSDL
	Port Name	Name of the selected port of a selected service (that you provide in the Service Name field) contained in the referenced WSDL . Same port names across receivers is not supported. To use the the same port names, you need to create a copy of the WSDL and use it.
	Operation Name	Name of the operation of selected service (that you provide in the Service Name field) contained in the referenced WSDL.
	Request Timeout	Specifies the time (in milliseconds) that the client will wait for a response before the connection is being interrupted. The default value is 60000 miliseconds (1 minute).
	Compress Message	Enables the WS endpoint to send compressed request messages to the WS Provider and to indicate the WS Provider that it can handle compressed response messages.
	Allow Chunking	Used for enabling HTTP chunking of data while sending messages.
	Connect without Client Authentication	Allows you to connect anonymously to the receiver system. Select this option if your server allows connections without authentication at the transport level.

Section	Parameters	Desciption
	Connect using Basic Authentication	Select this option to allow the ESB bus to connect to the receiver system using the deployed basic authentication credentials. When you select this option, you need to provide the credential name. *Credential Name*: Enter the credential name of the username-password pair specified during the deployment of basic authentication credentials on the cluster.
	Private Key Alias	Allows you to enter the private key alias name that gets the private key from the keystore and authenticates you to the receiver in an HTTPs communication. If you have selected the option of *Connect using Basic Authentication*, this field is not visible.
WS-Security (Web Services Security)	*WS-Security Configuration*	Specifies the way how WS-Security settings are to be configured. ○ *Via Manual Configuration in Channel* The security settings are manually to be configured (see below listed attributes). ○ *Based on Policies in WSDL* The security settings are specified as part of the receiver endpoint (within the endpoint WSDL) in elements as defined by the WS-Policy standard. ○ *None* No WS-Security is applied for message exchange.
	Credential Name (only configurable when for *WS-Security Configuration* the option **`Based on Policies in WSDL`** is selected)	

Section	Parameters	Desciption
	Username Token	Specifies the way how Username Token is to be configured. Select this option to authenticate at message level based on a user ID and a password. ○ *Plain Text Password* The Password is transfered in plain text (https encrypted). ○ *Hashed Password (Password Digest)* Cryptographic hash function is applied for password. ○ *None* No User Name Token is applied. If you have selected the option *Plain Text Password* or *Hashed Password (Password Digest)*, enter the credential name, that is, the alias that was assigned to the authorized user and password during tenant deployment.
	WS-Security Type	Specifies the combination of message protection methods that are to be applied. There are the following options: ○ *Sign Message* Tenant signs the payload. ○ *Sign and Encrypt Message* Tenant signs and encrypts the payload. ○ *None* No WS-Security Type is applied.
	Set Time Stamp (only configurable when for *WS-Security Configuration* the option **`Via Manual Configuration in Channel`** is selected)	Select this option to send a time stamp along with the message. In case a request-response pattern is configured, a time stamp is expected in the response message.
	Sender is Basic Security Profile Compliant	
	Private Key Alias for Signing	Specify an alias for the tenant private key that is to be used to sign the message. The tenant private key is used to sign the request message (that is sent to the WS provider (receiver)). The tenant private key has to be part of the tenant keystore.

Section	Parameters	Desciption
	Public Key Alias for Encryption (only configurable when for *WS-Security Type* the option **`Sign and Encrypt Message`** is selected)	Specify an alias for the public key that is to be used to encrypt the message. The receiver (WS provider) public key is used to encrypt the request message (that is sent to the receiver). This key has to be part of the tenant keystore.
	Initiator Token (only configurable when for *WS-Security Configuration* the option **`Via Manual Configuration in Channel`** is selected)	These entries define policies for the WS consumer. ○ *Include Strategy* Value **`Always to Recipient`** ensures that the WS consumer sends the certificate along with the message. ○ *X509 Token Assertion* Defines the format of the certificate being sent by the WS consumer along with the message.
	Recipient Token	These entries define policies for the WS consumer. *Include Strategy* Value **`Always to Initiator`** adds the certificate to the response message.
	Algorithm Suite Assertion	Specifies which algorithms are used by the WS consumer

5. Save the configurations in both the sender and receiver channel editors.
In the *Model Configuration* editor, when you place the cursor on the sender or receiver message flows, you can see the *SOAP Address* and *WSDL* information.

Related Information

WS-Security Configuration for the Sender SOAP 1.x Adapter [page 34]
WS-Security Configuration for the Receiver SOAP 1.x Adapter [page 35]

2.2.6.2.3.1 WS-Security Configuration for the Sender SOAP 1.x Adapter

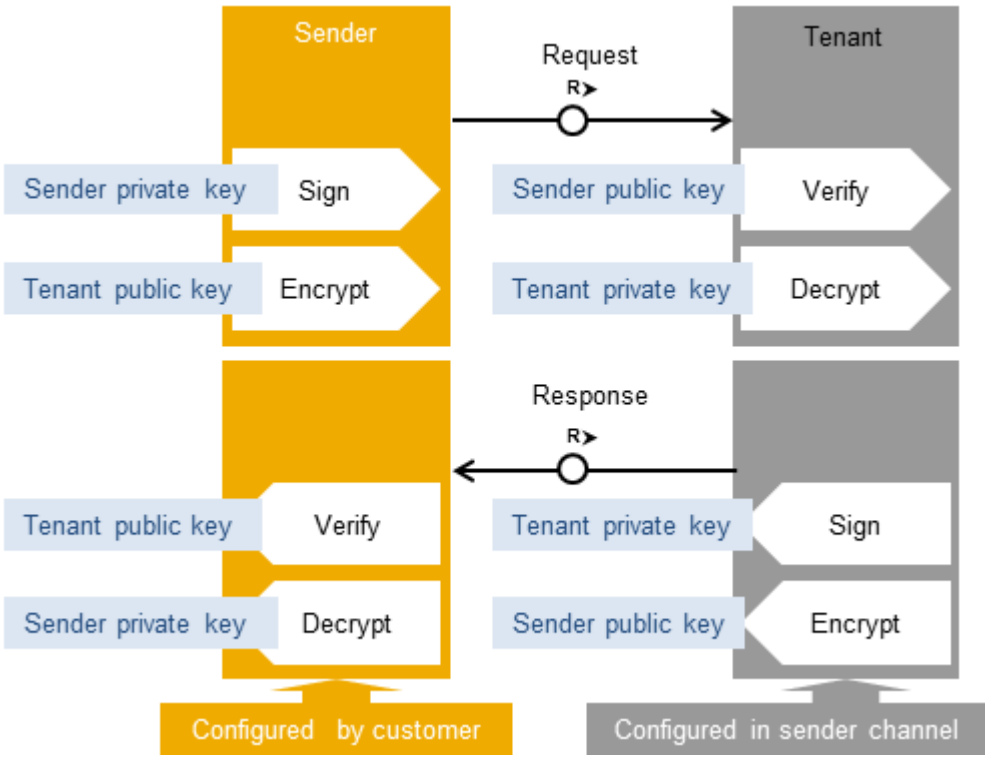
You use a sender channel to configure how inbound messages are to be treated at the tenant's side of the communication.

With regard to WS-Security in a sender channel, you specify the following:

- How the tenant verifies the payload of an incoming message (signed by the sender)

- How the tenant decrypts the payload of an incoming message (encrypted by the sender)

The following figure illustrates the setup of components:



The sender SOAP 1.x adapter allows the following combination of message-level security options:

- Verifying a payload
- Verifying and decrypting a payload

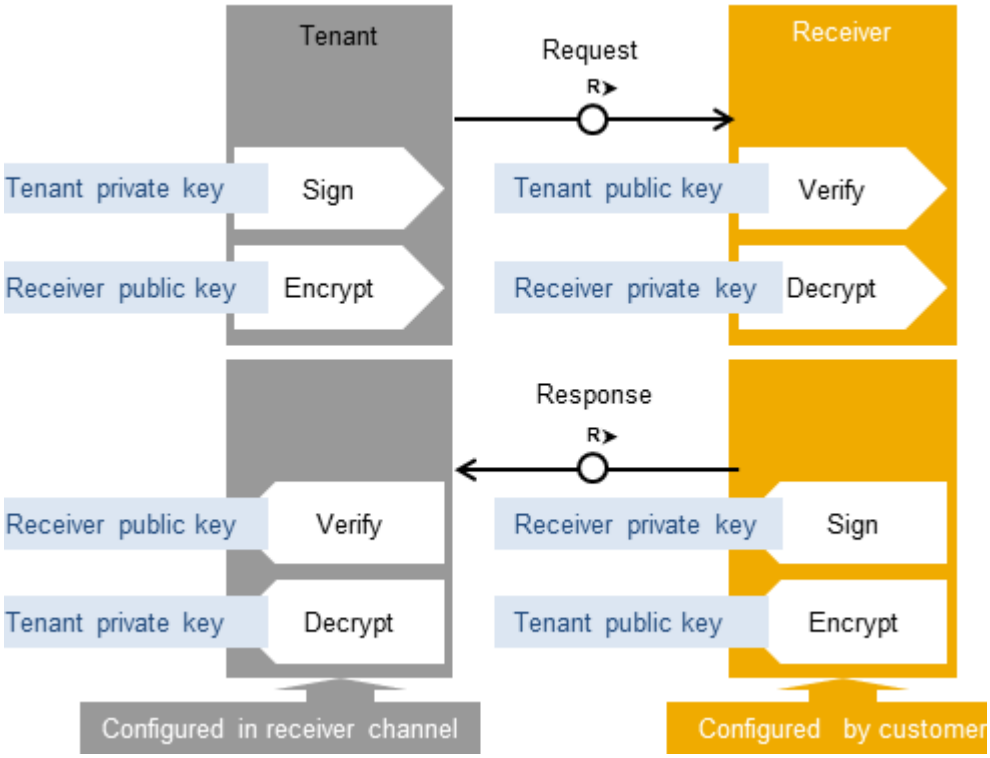
2.2.6.2.3.2 WS-Security Configuration for the Receiver SOAP 1.x Adapter

With a receiver channel you configure the outbound communication at the tenant's side of the communication.

With regard to WS-Security in a sender channel you specify the following:

- How the tenant signs the payload of a message (to be verified by the receiver)
- How the tenant encrypts the payload of a message (to be decrypted by the receiver)

The following figure illustrates the setup of components.



The receiver SOAP 1.x adapter allows to configure the following combinations of message security methods:

- Signing a payload
- Signing and encrypting a payload

Configuration Options for WS-Security

Signing and encryption (and verifying and decryption) is based on a specific set up of keys as illustrated in the figures. Moreover, for the message exchange, specific communication rules apply as been aggreed between Web service client and Web service provider (for example, if certificates are to be sent with the message).

There are two options how these security and communication settings can be specified:

- Based on Policies in WSDL
  Using this option, the security settings are specified as part of the receiver endpoint (within the endpoint WSDL) in elements as defined by the WS-Policy standard
  That way you can specify, for example, within the WSDL that certificates for message level security are sent with the message.
  For more information on the WS-Policy standard, see: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.html .
- Manual Configuration in Channel
  Using this option, you specify the required settings in the channel.
  The naming of the available attributes corresponds to the terminology used in the WS-Policy specification.
  If you use manual configuration, a sub set of the options as defined by the standard is supported. For more information on the standard, see http://www.w3.org/TR/ws-policy/ and http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf .

2.2.6.2.4 Configuring a Channel with HTTP Adapter

The HTTP adapter allows you to configure an outbound HTTP connection from SAP HANA Cloud Integration to a receiver.

Context

You can only configure a channel with the HTTP adapter type for outbound calls (from the tenant to a receiver system).

> **Note**
>
> If you want to dynamically override the configuration of the adapter, you can set the following headers before calling the HTTP adapter:

Table 8: Headers

Header	Description
CamelHttpUri	Overrides the existing URI set directly in the endpoint. This header can be used to dynamically change the URI to be called.
CamelHttpQuery	Overrides the existing URI parameters set directly in the endpoint. This header can be used to dynamically change the URI to be called.
Exchange.CONTENT_TYPE	HTTP content type that fits to the message (such as text/html)
Exchange.CONTENT_ENCODING	HTTP content encoding that fits the encoding during message transport (such as gzip)

Procedure

1. Choose the *General* tab page and, in the *Adapter Type* section, select *HTTP* from the *Adapter Type* dropdown list.
2. Choose the *Adapter Specific* tab page and enter the values for the available parameters. The table below gives a description of the fields and the possible values you can enter:

Table 9: Parameters and Values for the Receiver HTTP Adapter

Parameter	Description
Address	URL of the target system (receiver), for example, `https://mysystem.com`. Note that the authentication method `Client Certificate` requires the HTTPS protocol. For `Basic authentication` it is strongly recommended that you use the HTTPS protocol. If you have selected one of these authentication methods, you therefore have to enter an `https` URL. You can also specify HTTP parameters in the URL. However, if you select the HTTP method `POST`, parameters are usually sent in the body. You therefore get a warning message if you configure this parameter-value combination. The following URL parameters are currently not allowed for technical reasons: ○ throwExceptionOnFailure ○ bridgeEndpoint ○ transferException ○ client ○ clientConfig ○ binding ○ sslContextParameters ○ bufferSize This parameter can be externalized.
Query	You can specify a query string to be transferred with the HTTP request. Query strings must not be entered in the *Address* field. Note that the query string has to start with the character `&` (ampersand). This parameter can be externalized.

Parameter	Description
HTTP Method	Defines the action to be performed by the HTTP request. You can select one of the following values: ○ POST Requests that the receiver accepts the data enclosed in the request body. ○ GET Sends a GET request to the receiver. ○ PUT Updates or creates the enclosed data on the receiver side.
Authentication Method	Defines how SAP HCI (as the HTTP client) will authenticate itself against the receiver. You can select one of the following authentication methods: ○ None ○ Basic SAP HCI authenticates itself against the receiver using user credentials (user name and password). It is a prerequisite that user credentials are specified in a Basic Authentication artifact and deployed on the related tenant using the Integration Operations feature. ○ Client Certificate SAP HCI authenticates itself against the receiver using a client certificate. It is a prerequisite that the required key pair is installed and added to a keystore. This keystore has to be deployed on the related tenant using the Integration Operations feature. The receiver side has to be configured appropriately.
Credential Name (only if the Basic authentication option is selected)	Identifies the Basic Authentication artifact that contains the credentials.
Private Key Alias (only if the Client Certificate authentication option is selected)	Identifies the related key pair within the SAP HCI client keystore.

Parameter	Description
Request Timeout	Specifies the time (in milliseconds) that the client will wait for a response before the connection is interrupted. The default value is 60000 milliseconds (1 minute).

3. Save the configuration in the receiver channel editor.

2.2.6.2.5 Configuring a Channel with SFTP Adapter

The SFTP adapter uses the SSH protocol to transfer files.

Context

Unlike the standard FTP, the SFTP adapter uses a certificate and keystore to authenticate the file transfer. The SFTP connector achieves secure transfer by encrypting sensitive information before transmitting it on the network.

Procedure

1. Choose the *General* tab page and, in the *Adapter Type* section, select *SFTP* from the *Adapter Type* dropdown list.
2. Choose the *Adapter-Specific* tab page and enter values for the fields on the *Source*, *Processing*, and *Advanced* tabs. The table below contains descriptions of the fields and the possible values you can enter:

Table 10: Parameters and Values of Sender SFTP Adapter

Tab	Sections	Parameters	Description
Source	File Access Parameters	Directory	File path from where the file should be read, for example, <dir>/<subdir>.
		File Name	Name of the file to be read. **Note** If you do not enter a file name and the parameter remains blank, all the files in the specified directory are read. Regular expressions, such as ab*, a.*, *a*, and so on are not supported.
	Connection Parameters	Server Host	Host name or IP address of the SFTP server and an optional port, for example, wdfd00213123:22.

Tab	Sections	Parameters	Description
		User Name	ID of the user performing the file transfer.
		Connection Timeout (in ms)	Maximum time to wait for the SFTP server to be contacted while establishing connection or performing a read operation. Default value: 10000 ms
		Maximum Reconnect Attempts	Maximum number of attempts allowed to reconnect to the SFTP server. Default value: 3 Use 0 to disable this behavior.
		Reconnect Delay	Length of time to wait before attempting to reconnect to the remote SFTP server. Default value: 1000 ms
		Disconnect After Processing	Disconnect from the FTP server after each message processing.
Processing	Processing Parameters	Read Lock Strategy	Prevents files that are in the process of being written from being read from the SFTP server. The endpoint waits until it has an exclusive read lock on a file before reading it. Select one of the following options based on the capabilities of the SFTP server: ○ None: Does not use a read lock, which means that the endpoint can immediately read the file. *None* is the simplest option if the SFTP server guarantees that a file only becomes visible on the server once it is completely written. ○ Rename: Renames the file before reading. The *Rename* option allows clients to rename files on the SFTP server. ○ Content Change: Monitors changes in the file length/modification timestamp to determine if the write operation on the file is complete and the file is ready to be read. The *Content Change* option waits for at least one second until there are no more file changes. Therefore, if you select this option, files cannot be read as quickly as with the other two options.
		Traverse Directories Stepwise	Changes directory levels one at a time.

Tab	Sections	Parameters	Description
		Recursive	Allows you to look for files in all the subdirectories of the directory.

Tab	Sections	Parameters	Description
		Action After Processing	Allows you to specify how files are to be handled after processing. You can select one of the following options: ○ *Delete File* The file is deleted after it has been read. ○ *Keep File and Mark as Processed in Idempotent Repository* Enables an idempotent repository to prevent a file from being consumed twice. Select this option for SFTP servers that do not allow deletion or moving of files, but the files are to be read only once. You can select one of the following idempotent repository options: ○ *In Memory*: Keeps the file names in the memory. Files are read again from the SFTP server when the runtime node is restarted. **Note** If there is a system crash or disaster, this data will be lost and cannot be recovered later. ○ *Database*: Stores the file names in a database to prevent the files from being read again when the runtime node is restarted. File name entries are deleted by default after 90 days. **Note** The database uses the server name, directory, and file name as key values to identify files uniquely across integration flows of a tenant. ○ *Keep File and Process Again* The file is kept on the SFTP server and file processing is repeated. You can use this option, for example, for testing purposes. ○ *Move File* The file is moved to another directory. If you select this option, you need to specify the target directory.

Tab	Sections	Parameters	Description
			You can specify the target directory dynamically, for example, using the timestamp of the message. The following example uses backup folders with timestamps and replaces the file extension with `bak`: `backup/${date:now:yyyyMMdd}/${file:name.noext}.bak`
		Alert Threshold for Retry	If the number of attempts to retry polling a message from the SFTP server exceeds this threshold value, an alert is raised. The default value '0' indicates that the alert is not raised. **Note** If two or more sender channels are configured with the SFTP connector, the value for the *Alert Threshold for Retry* parameter should be the same.
	Delay Parameters	Initial Delay (ms)	Delay before the polling of files starts. Default: 1000 ms
		Delay (ms)	Delay before the next polling of files starts. Default: 500 ms
		Use Fixed Delay	Controls whether a fixed delay or fixed rate is used.
Advanced	Advanced Parameters	Buffer Size (in KB)	Write file content using the specified buffer size. Default: 128 KB
		Run Logging Level	Logs a start/complete log line on polling. Default: TRACE

Tab	Sections	Parameters	Description
		Maximum Messages per Poll	Maximum number of messages to gather per poll. Default: 0 Example: 1000 can be set as a limit. **Note** When you use the sender SFTP adapter in combination with an Aggregator step and you expect a high message load, consider the following recommendation: Set the value for *Maximum Messages per Poll* to a small number larger than 0 (for example, **20**). This ensures a proper message processing status logging at runtime.
		Flatten File Names	Flatten the file path by removing the directory levels so that only the file names are considered and they are written under a single directory.

Table 11: Parameters and Values of Receiver SFTP Adapter

Tab	Sections	Parameters	Description
Target	File Access Parameters	Directory	File path to which the file should be written, for example, <dir>/<subdir>.
		File Name	Name of the file to be written.
	Connection Parameters	Server Host	Host name or IP address of the FTP server.
		User Name	ID of the user performing the file transfer.
		Connection Timeout (in ms)	Maximum time (in ms) to wait for the FTP server to be contacted while establishing connection or performing a read operation. Default value: 10000 ms
		Maximum Reconnect Attempts	Maximum number of attempts allowed to reconnect to the FTP server. Default value: 3 Use 0 to disable this behavior.
		Reconnect Delay	Length of time to wait before attempting to reconnect to the remote FTP server. Default value: 1000 ms

Tab	Sections	Parameters	Description
		Disconnect After Processing	Disconnect from the FTP server after each message processing.
Processing	Processing Parameters	Traverse Directories Stepwise	Changes directory levels one at a time.
		Automatically Create Directories	Automatically creates missing directory levels as provided in the file's path name.
		File Exist	If the file already exists in the target, allow the following: ○ Override: Replace the existing file content with the new one. ○ Append: Add the new file content to the end of the existing one. ○ Fail: Do not perform any action and raise a failure. ○ Ignore: Do not perform any action.
Advanced	Advanced Parameters	Buffer Size (in KB)	Write file content using the specified buffer size. Default: 128 KB
		Run Logging Level	Logs a start/complete log line on polling. Default: TRACE
		Flatten File Names	Flatten the file path by removing the directory levels so that only the file names are considered and they are written under a single directory.
		Prevent Backward Path Traversal	If the file contains any backward path traversals such as \..\ or /../.. , this can lead to a potential risk of directory traversal. In such a case, this parameter autogenerates a file name based on the unique message ID and replaces the header with the unique message ID. The unique message ID is logged in the Message Processing Log. **Recommendation** We recommend that you specify the *Directory* and *File Name* fields to avoid any security risks. If you provide these fields, the header is not considered.

3. Save the configurations in both the sender and receiver channel editors.
   In the *Model Configuration* editor, when you place the cursor on the sender or receiver message flows, you can see the *Server Host* and *Directory* information.

Next Steps

> **Note**
>
> SFTP polling is supported in the following way: The same file can be polled by multiple endpoints configured to use the SFTP channel. This means that you can now deploy an integration flow with a configured SFTP channel on multiple runtime nodes (which might be necessary to meet failover requirements) without the risk of creating duplicates by polling the same file multiple times. Note that to enable the new option, integration flows (configured to use SFTP channels) that have been developed prior to the introduction of this feature have to be regenerated.
>
> This feature was introduced with the release of the Integration Designer in 03 2014.

2.2.6.2.6 Configuring a Channel with SuccessFactors Adapter

Prerequisites

You have created an integration project and an integration flow.

To successfully run the **Operations Modeler**, your Java Virtual Machine (JVM) must contain the security certificate recommended by the SuccessFactors system. **Example:** VeriSign Class 3 Public Primary Certification Authority - G5 security certificate.

> **Note**
>
> First, you must verify if the JVM contains the security certificate that is used by SuccessFactors system. If not, then download the certificate from the appropriate security certificate vendor and install it. You can refer to JVM documentation for verifying and installing the security certificate on to your JVM. Ensure that the IP addresses of the HCI run-time worker node and the systems you are using to connect to the SuccessFactors system are in the list of allowed IP addresses.

Context

The SuccessFactors adapter provides three message protocols for you to communicate with the SuccessFactors system. They are:

1. *SOAP*
2. *OData V2*

> **Note**
>
> This is available only for the receiver channel.

3. *REST*

You can choose the protocol you want based on the scenario you want to execute.

You need to provide the following details in order to communicate with the SuccessFactors system.

- **Connection details** – Details required to establish a connection with the SuccessFactors system
- **Processing details** – Information required to process your modeled operation
- **Scheduler** – Settings that enable you to schedule a data polling cycle at regular intervals

> **Note**
>
> The scheduler is only applicable for the sender channel.

Procedure

You use this procedure to configure a channel with SuccessFactors adapter.

1. Double-click the channel that you want to configure in the *Model Configuration* tab page.
2. Choose the *General* tab page and choose *Browse* in the *Adapter Type* screen area.
3. Select *SuccessFactors* in the *Choose Adapter* window and choose *OK*.
4. If you want to use *SOAP* based Webservice from SuccessFactors, perform the following substeps:
   a. Choose *SOAP* in *Message Protocol* field.
   b. Choose the *Adapter Specific* tab page and enter values based on the description given in the table.

Table 12: Parameters & Values of SuccessFactors Adapter for SOAP

Section	Parameter	Description
Connection Details	Address	URL of the SuccessFactors data center that you would like to connect to. **Example:**https://sales-demo4.successfactors.com
	URL Suffix	This field is automatically filled based on the protocol you choose. For *SOAP*, the value is **/sfapi/v1/soap**.

Section	Parameter	Description
	Credential Name	Name of the system you use to establish the connection. You use the same name to deploy credentials on the cluster using *Basic Authentication*. Refer to operations guide for details on deploying credentials on the cluster.
Processing Details	Call Type	Type of call that HCI system makes to the SuccessFactors system. Choose *Synchronous Query* to perform normal operations. Choose *Adhoc Asynchronous Query* to perform adhoc operations
	Operation Details	*Entity:* The Entity used to fetch data from the system. *Operation:* Query/Insert/Update/Upsert *Query/Fields:* SFQL (SuccessFactors Query language) Query Type that is used to communicate with SuccessFactors system. **Note** Query is the only operation available in sender channel. The Model Operation button launches the SuccessFactors Query Language (SFQL) editor. Refer to section SuccessFactors Operations Modeler for more details

Section	Parameter	Description
	Page Size	This field is applicable only in case of *Query* operation. This indicates the number of records that the HCI system reads from the SuccessFactors system when the *Operation* is executed. If you find that the *Operation* is not executing due to a time-out, reduce the *Page Size* and execute the operation again.

5. If you want to use the OData services from SuccessFactors, perform the following substeps:
   a. Choose *ODataV2* in *Message Protocol* field.
   b. Choose the *Adapter Specific* tab page and enter values for fields based on the description given in the table.

Table 13: Parameters & Values of SuccessFactors Adapter for ODataV2

Section	Parameter	Description
Connection Details	Address	URL of the SuccessFactors data center that you would like to connect to. **Example:**https://sales-demo4.successfactors.com
	URL Suffix	This field is automatically filled based on the protocol you choose. For *ODataV2*, the value is **/odata/v2**.
	Credential Name	Name of the system you use to establish the connection. You use the same name to deploy credentials on the cluster using *Basic Authentication*. Refer to operations guide for details on deploying credentials on the cluster.

Section	Parameter	Description
Processing Details	Operation Details	*Operation:* Query /Read/Insert/ Update *ResourcePath:* URI that is used that is used to communicate with SuccessFactors system for the OData operation *Path to edmx:* The location where the EDMX file is downloaded

6. If you want to use *REST* message protocol, perform the following substeps:

   REST message protocol is available only for the Learning Management Solution (LMS).

   a. Choose *REST* in the *Message Protocol* field.
   b. Choose the *Adapter Specific* tab page and enter values for fields based on the description given in the table.

   Table 14: Parameters & Values of SuccessFactors Adapter for REST

Section	Parameter	Description
Connection Details	Credential Name	Name of the system you use to establish the connection. You use the same name to deploy credentials on the cluster using *OAuth2 Authentication*. Refer to operations guide for details on deploying credentials on the cluster.
Processing	Address	URL of the LMS REST service. **Example:** <<server>>/learning/public-api/rest
	Resource Path	Resource Path of the REST service. **Example:** v1/current-user/curricula
	Operation	Operation for the REST service. **Note** Only *GET* and *POST* operations are supported currently.

Section	Parameter	Description
	Parameters	Parameter to be sent to the REST service. **Example:** creationDate=1&active=true **Note** In case of the GET operation you can fetch just the modified records in subsequent runs by using the condition **lastModifiedDate=${deltasync.maxDateFromLastRun}**.
	Page Size	The number of records that are read from SuccessFactors system in one request If you find that the *Operation* is not executing due to a time-out, try executing the *Operation* by reducing the *Page Size*.

7. If you are configuring the sender channel, perform the following substeps to configure the scheduler:
   a. Choose the *Scheduler* tab page.
   b. Enter the scheduler details based on the description given in the table below.

Table 15: Parameters & Values of SuccessFactors Adapter Scheduler

Field	Description
Run once	Runs a data polling process immediately after saving the configuration
On Date	Specific date on which the data polling process has to be initiated to fetch data from the SuccessFactors system
Daily	Run message polling every day to fetch data from the SuccessFactors system
Weekly	Run the message polling every week on specified days of the week to fetch data from the SuccessFactors system.

Field	Description
Monthly on Day	Execute the message polling every month on the specified date to fetch data from the SuccessFactors server. **Note** If the specified date is not applicable to a month, the data polling is not executed in that specific month. For example, if 30th day is selected in the month of February, polling is not executed as 30th is not a valid day for February.
Time	The time at which the data polling cycle has to be initiated. For example, if you want the data polling to be started at 4.1PM, enter 16:10. Note that the time must be entered in 24-hour format.
Every xx minutes between HH hours and HH hours	The connector fetches data from the SuccessFactors system, every 'xx' minutes between HH hours and HH hours. **Note** If you want the polling to run for the entire day, enter 1 and 59.
Every xx hours between HH hours and HH hours	The connector fetched data from the SuccessFactors system, every 'xx' minutes between HH hours and HH hours. **Note** If you want the polling to run for the entire day, enter 1 to 23.
Time Zone	Select the Time Zone that you want to use as reference for scheduling the data polling cycle.

8. Save the changes.

> **Note**
>
> The password for connecting to the SuccessFactors system should be deployed onto the tenant via the 'Credentials' deployment wizard available in the Node Explorer.

> ⚠ Restriction
>
> The SuccessFactors Adapter uses polling as a mechanism to fetch data from the SuccessFactors system. An IFLMAP based integration flow, when deployed in HANA Cloud Integration, deploys in multiple IFLMAP worker nodes. Polling is triggered from only one of the worker nodes. The message monitoring currently displays the process status from the worker nodes where the scheduler is not started. This results in the message monitor displaying messages with status less than 10ms, from the worker nodes, where the schedule was not triggered. You can ignore the monitoring statuses.

Modeling SFQL Operations for connecting to SuccessFactors SOAP WebService

Prerequisites

You have configured the sender or receiver channel of the SuccessFactors adapter.

Context

You use the SuccessFactors *Model Operation* feature to select the required entity, choose an operation and specify the fields based on your business requirements. You can create SFQL (SuccessFactors Query language) based fields list that is used to execute scenarios. You can also fetch the updated data after your last successful data fetch run. This is called as 'Delta Sync'. Refer to Delta Sync scenarios for more details.

You use this procedure to model a SFQL query.

Procedure

1. Choose the *Adapter Specific* tab page in the sender channel configuration.
2. Choose *Model Operation* in the *Processing* tab page.
3. Connect to the SuccessFactors system.

   Use the following parameters to connect to the SuccessFactors test or development environment to fetch the SuccessFactors API details and choose *Next*.

Field	Data to be Filled
Address	URL used to connect to the SuccessFactors API in Development or Test Landscape.
Company ID	SuccessFactors Company ID
User Name	User name to be used to connect to the systems

Field	Data to be Filled
Password	Password to be used to connect to the system
Proxy Host	Proxy Host address of the proxy setting to be used to connect to the SuccessFactors System
Proxy Port	Proxy Port of proxy setting to be used to connect to the SuccessFactors System

4. Select the *Entity*.

Parameter	Description
Entity List	List of the available Entities. Data related to these entities is fetched.

5. Select the *Operation* and *Fields* of the entity.

   Choose the attributes (fields) you would like to fetch from the entity you selected in the previous step.

Parameter	Description
Operation	Select the operation that you want to perform. **Note** Only *Query* operation is available in the sender channel.
Fields	Select the fields related to the entity that should be used in the operation.

6. Configure filter conditions (optional step)

   This step is available only in case of 'Query' Operation. In this step, you can configure the filter condition that you wish to apply to the fields you would like to fetch from the selected entity.

Field	Description
Filter Field	Field that is used in the SuccessFactors API 'WHERE' clause for filtering. **Note** Field set contains the set of filterable fields returned from the SuccessFactors API that you can use in the Filter Condition
Operation	Operator to be used in the WHERE condition Example < , >

Field	Description
Type	Value that the filter filed has to be compared against. When the Type is 'Text' then specify the exact value When the Type is 'XPath' specify the entire XPath value. In case the previous Integration flow step is a content enricher, the XPath must be specified as a relative path, starting with double-slash '\\'. When the Type is 'Delta Sync' the value is populated with maxDateFromLastRun.
Value	Value that the filter filed has to be compared against. When the Type is 'Text' then specify the exact value When the Type is 'XPath' specify the entire XPath value. In case the previous Integration flow step is a content enricher, the XPath must be specified as a relative path, starting with double-slash '\\'. When the Type is 'Delta Sync' the value is populated with maxDateFromLastRun.
Condition	'AND' or 'OR' condition that needs to be used in the Query WHERE clause filter condition.
Add	The condition will be added to the generated SuccessFactors Query **Note** Multiple conditions can be added if required
Remove	Any condition that is already added to the list can be selected and removed from the final SuccessFactors Query

You can configure the filter condition to execute delta sync scenarios. Refer to **Configuring Delta Sync Scenarios** for more details.

7. Configure sorting conditions (optional step)

   Choose the attributes (fields) you would like to fetch from the selected SuccessFactors Entity.

Field	Description
Order By	Field that is used in the SuccessFactors API 'ORDER BY' clause for sorting. **Note** Field set contains the set of sortable fields returned from the SuccessFactors API that you can use in the 'ORDER BY' condition
Descending	The list is sorted in descending order if this button is selected. **Note** If not selected, the data is ordered in Ascending order by default
Add	The condition will be added to the generated SuccessFactors Query **Note** Multiple conditions can be added if required
Remove	Any condition that is already added to the list can be selected and removed from the final SuccessFactors Query

8. Choose *Finish*

   The 'Finish' button is activated only if you select some fields of the entity in step 3. When you choose *Finish*, the system creates a XML schema file with the selected entities. You can access the schema file in src.main.resources.wsdl folder of your project. If there is an existing XML schema file, you have the option of overwriting the existing file or creating a new file after choosing the finish option. This file can be used in the integration flow like mapping step.

   One of the root elements in the XML schema file is the *Entity Name*. In cases where the *Entity Name* is in the format **<EntityName>_$XX**, only **<EntityName>** is used as the root element of the XML schema file. **$XX** is dropped from the root element name of the XML schema so that you can use the same integration flow in other SuccessFactors company ID without changing the mapping.

Configuring Delta Sync Scenarios

You can configure SuccessFactors connector to fetch the modified or delta records instead of fetching all the records. This optimizes the polling mechanism. This is known as delta sync configuration.

Note

You have configured the sender channel of SuccessFactors connector.

If you wish to add more filter conditions after configuring for delta sync, use the appropriate operators and add them. Once the query is executed, the relevant scenarios are executed.

> **Note**
>
> The following steps guide you to configure the delta sync conditions only. Refer to **Modelling Operations** for end-to-end procedure on creating and executing operations.

Delta Sync

With this configuration, the system fetches all records from the beginning of time (1/1/19070, by default) in the first run. Only modified records are fetched in the subsequent runs.

1. In *Configure Filter Condition for Fields* window, select a field of type DATETIME for the *Filter Field*. Example: *lastModified*
2. In the *Operation* field, select >.
3. In *Type* field, select *Delta Sync*. *maxDateFromLastRun* is automatically populated in the *Value* field.

> **Note**
>
> If the payload from the SuccessFactors system has *execution_timestamp* as one of the fields, that time-stamp is used as the reference date for the subsequent delta sync polling cycles. The date specified in the *Query* is ignored.

Modify Query in Existing Delta Sync Configuration

With this configuration, you can modify the query in an existing delta sync configuration. The system will consider the new Query and fetch only modified records in the subsequent polling cycles.

1. Add or remove the new fields that you wish to fetch in *Model Operation* window.
2. If you want to add new filter conditions. In the *Configure Filter Conditions for Fields* window.
3. You can also modify or remove existing filter conditions in the *Configure Filter Conditions for Fields* window.
4. Continue with the existing delta sync configuration.

Reset Existing Delta Sync Configuration

You perform these steps only to reset an existing delta sync configuration. After reset, the configuration enables you to fetch data from the beginning of time (1/1/1970) in the first polling cycle and fetch only modified records in the subsequent polling cycles.

1. In the channel configuration, enter a new channel name in the *Channel Details* section. The new name resets the existing delta sync configuration.

> ⚠ Caution
>
> Choose a unique channel name. Do not use names that were used in earlier delta sync configurations.

2. Save the configuration.

Fetch Records after a Specified Date in the First Run and Fetch Modified Records in Subsequent Runs

With this configuration, you can specify a date that will be used as a reference to fetch records. The system fetches the records that are modified or added after the specified date in the first polling cycle. The modified records are fetched in the subsequent polling cycles.

1. Select the fields that you want to fetch in the *Model Configuration* window.
2. In *Configure Filter Condition for Fields* window, select a DATETIME type field in *Filter Field* window. Example: *lastModified*.
3. In *Operation* field, choose >=.
4. In the *Value* field, enter the date after which you want the records to be fetched from the system.
5. Choose *Add*.
6. Select a field of type DATETIME for the *Filter Field*. Example: *lastModified*.
7. In the *Operation* field, select >.
8. In *Type* field, select *Delta Sync*. *maxDateFromLastRun* is automatically populated in the *Value* field.
9. In the *Operators* field, choose *AND*.

Modeling OData Operations to Connect to SuccessFactors OData Service

Prerequisites

You must have configured the receiver channel of the SuccessFactors adapter.

Context

If you have chosen *ODatav2* protocol to connect to the SuccessFactors system, use this procedure to model a SFQL query.

Procedure

1. Choose *Model Operation* in the *Adapter Specific* tab page.
2. If you want to use a local EDMX file to connect to the system, perform the following substeps:
   a. Select the *Local EDMX File* checkbox.
   b. Choose *Browse*.
   c. Select an EDMX file in the *EDMX Selection* window and choose *OK*.
3. To connect to the SuccessFactors OData service, enter values in fields based on the description given in the table below and choose *Next*.

Field	Description
Address	The URL of the SuccessFactors OData Service provider
Company ID	Company ID that you are using to connect to the SuccessFactors system
Username	Username for authentication
Password	Password for authentication
Proxy Host	The proxy host that you are using to connect to the SuccessFactors system
Proxy Port	The port of the proxy host that the system uses to establish the connection with the SuccessFactors OData service provider

> **Note**
>
> The proxy settings are enabled only if you select the *Enable proxy communication* checkbox.

4. Select the *Entity* in *Select Entity for an operation* window and choose *Next*.
5. Choose the *Navigation Depth* from the dropdown list.

> **Note**
>
> The Navigation Depth is the level up to which you want to view the entity association. For example, consider that entity B is associated with entity A and entity C is associated with entity B. If you choose entity A in the Select Entity for an operation window and choose Navigation Depth as 1, you can navigate till entity B. If you choose Navigation Depth as 2, you can navigate up to entity C.

6. Choose the *Operation* from the dropdown list based on the description given in the table.

Field	Description
Query (GET)	Used to fetch data from the OData service
Update (PUT)	Used to update data to an OData service
Insert (POST)	Used to insert data to an OData service
Read (GET)	Used to fetch a unique entity from the OData Service. Passes the key fields along with the Entity in the URI (Universal Resource Indicator). **Format:**<Entity>(Keyfield 1, Keyfield 2, etc.)

7. Select the required fields for the operation from the *Fields* screen area and choose *Next*.

> **Remember**
>
> If you choose *PUT* or *POST* operation, this is the last step. Choose *Finish*.

8. Enter values in the *Top* and *Skip* fields based on the description given in table below. This is applicable only in case of *Query* operation

Field	Description
Top	If you enter value 'x', only the top 'x' values are fetched from the OData service provider
Skip	If you enter value 'x', the top 'x' values are ignored and the remaining records are fetched from the OData service provider

9. Select the values based on the description given in the table below to add filter conditions to the operation. The filter step is available only in case of the Query (GET) Operation.

Field	Description
Filter Field	The field that you are using to set a filter condition
Operation	The operation that you are using to set the filter condition. Example: >, =
Input	The input type of the value that you are adding to the filter condition
Value	The value you are using to set the filter condition
Operators	The operator that you are using to add the filter condition to the operation

10. Choose *Finish*.

Results

Choosing Finish generates an XSD and EDMX files.

The HCI enterprise service bus (ESD) processes data in the XSD format. You use this XSD file in mapping step for data transformation.

The EDMX file contains the OData entity specification from the OData service provider. You can use this file in the subsequent operation modeling steps to connect to the OData service provider.

2.2.6.2.7 Configuring a Channel with OData Adapter

Prerequisites

- You have created an integration project
- You have created an integration flow

Context

The OData connector enables you to connect to systems exposing OData services (OData service providers). The connector allows the integration developer to connect to an OData service and perform the operations Query (GET), Update (PUT), Insert (POST), Read (GET), allowing the querying, updating and inserting of data.

You use this procedure to configure the receiver channel of the OData connector.

Procedure

1. Double-click the receiver channel of the integration flow in the *Model Configuration* tab page.
2. Choose *Browse* in the *Adapter Type* screen area.
3. Choose *OData* in the *Choose Adapter* window and choose *OK*.
4. Choose the *Adapter Specific* tab page and enter details in fields based on the description given in the table below.

Field	Description
Address	URL of the OData service provider you want to connect to
Authentication Method	Select *Basic* from the dropdown list if you want to use basic authentication to connect to the OData service provider Select *Client Certificate* from the dropdown list if you want to use a certificate for authentication while connecting to the OData service provider. **⚠ Restriction** You cannot use client certificate for connecting to the OData service provider while modeling operations.

Field	Description
Operation Details	This contains details to the operation. This could be Query (GET), Update (PUT), Insert (POST), Read (GET) *ResourcePath:* This is the URI that is appended to the OData service endpoint when connecting to the service provider. For more information, see Modeling Operations for OData Adapter

5. Save the configuration.

Modeling Operations for OData Adapter

Prerequisites

You have configured the receiver channel of the OData adapter.

Context

You use the *Model Operation* feature in the OData adapter to model an operation [Query (GET), Update (PUT), Insert (POST), Read (GET)]. You also select the *ResourcePath*, the URI using which you transact with the OData service provider.

Procedure

You use this procedure to model an operation with the Odata adapter.

1. Double-click the OData receiver channel of the integration flow in the *Model Configuration* tab page.
2. Choose the *Adapter Specific* tab page.
3. Choose *Model Operation*.
4. If you want to use a local EDMX file to connect to the OData service provider, perform the following substeps:

> **Remember**
>
> If you have used client certificate for connecting to the OData service provider, you need to download the EDMX file from the OData service provider and manually import it to the *src.main.resources.edmx* folder. This enables you to use this option while modeling operations.

   a. Select the *Local EDMX File* checkbox.

b. Choose *Browse*.
c. Select an EDMX file in the *EDMX Selection* window and choose *OK*.

5. If you want to enter connection details manually, enter values in fields based on the description given in the table.

Field	Description
Address	URL of OData service provider you are connecting to
Username	Username you are using for authentication
Password	Password you are using for authentication

> **Note**
>
> You cannot use this option if you have selected client certificate as the authentication method while configuring the channel.

6. Select the entity in *Select Entity for an Operation* window and choose *Next*.
7. Choose the *Operation* from the dropdown list based on the description given in the table.

Operation	Description
Query (GET)	Used to fetch data from the OData service
Update (PUT)	Used to update data to an OData service **Restriction** This operation is not supported for associated entities
Create (POST)	Used to insert data to an OData service
Merge (MERGE)	Used to merge data with existing data in OData service **Restriction** This operation is not supported for associated entities
Read (GET)	Used to fetch a unique entity from the OData Service. Passes the key fields along with the Entity in the URI. **Format:** <Entity>(keyfield 1, keyfield x,..)

8. Select the required fields for the operation form the *Fields* screen area.

> **Note**
>
> IN the case of Update (PUT) or Insert (POST) operation, this would be the last step. Choose *Finish*.

9. If you have chosen the operation as *Query (GET)*, enter values in *Top* and *Skip* fields based on the description given in table.

Field	Description
Top	If you enter a value 'x', the system fetches the top x records form the OData service provider

Field	Description
Skip	If you enter a value 'x', the system skips x records from top and fetches the remaining records from the OData service provider

10. Choose *Next*.
11. If you want to add filter conditions to the operation, enter values in fields based on the description given in table.

Field	Description
Filter Field	The field that you are using to set a filter condition
Operation	The operation that you are using to set the filter condition
Input	The input type of the value that you are adding to the filter condition
Value	The value you are using to set the filter condition
Operators	The operator that you are using to add the filter condition to the operation

12. Choose *Finish*.

Results

Choosing the *Finish* button generates an XSD and EDMX files.

The XSD is the format in which data is processed in the HCI esb. You use this xsd file in the mapping step for data transformation.

The EDMX file contains the OData Entity specification form the provider. This can be used when you model operation again by choosing ‘Local EDMX’ file.

Defining Mapping for OData Batch Processing

Prerequisites

- You have created an integration project and integration flow
- You have configured the OData adapter receiver channel
- You have modeled PUT or POST operation using the OData adapter
- You have imported the input payload XML format into the eclipse project

Context

When you choose batch processing for PUT and POST operations for the OData adapter, the payload format that is sent to the OData service must be in the recommended structure. You can use the input XSD that is generated when you model the operation with a mapping step to transform the payload into the recommended XSD structure. You can alternatively use XSLT or content modifier to do this.

Procedure

This procedure enables you to transform the input payload XML into the recommended batch processing structure.

1. Choose *File* > *New* > *Other*.
2. In the *New* wizard, choose *SAP HANA Cloud Integration* > *Message Mapping* and choose *Next*.
3. In the *General Details* section of the *New Message Mapping* window, enter *Name* and *Description*.
4. In the *Location Details* section of the *New Message Mapping* window, choose *Browse* and select the project that you are working in. Choose *Ok*.
5. In the *New Message Mapping* window, choose *Finish*.
6. In the *Source Element* section, choose *Add*.
7. In the Sel*ect a XSD or WSDL file* window, select the input payload format XML and choose *OK*.
8. In the *Target Element* section, choose *Add*.
9. In the *Select a XSD or WSDL file* window, select the XSD file that was generated when you modeled the operation and choose *OK*.
10. Choose the *Definition* tab page and perform the following substeps to map the elements in the payload XML to the target XSD.
    a. Map the *Entity Type* element in the left pane to the *batchChangeSet* on the right pane.
    b. Map the fields in the left pane to the same fields on the right pane.
    c. Map the *batchChangeSet*, *Entity Set* and *Entity Type* elements on the right pane except the headers to a constant. The value of the constant can be a dummy example value.
    d. Choose the method element and double-click *Constant* element in the properties view.
    e. In the *Constant Parameters* window, enter the operation (PUT/POST) in the *Value* field and choose *OK*.

       This is the same operation that you have chosen while modeling the OData operation.

> **Note**
>
> Refer to 2031746 for more details on the structure of request and response XSD.

2.2.6.3 Defining Message Transformation

2.2.6.3.1 Assigning Mapping

Prerequisites

- You have configured the connections to an On-Premise repository if you have to import mapping from that repository into the workspace.

  > **Note**
  >
  > For more information on setting the connections to the repository, see the task 'Setting the Connections to the External Repository' in Configuring the Tool Settings [page 5]. You can import message mappings (.mmap) from ES Repository with server version 7.1 onwards, and operation mapping from ES Repository with server version 7.3 EHP1 SP3 .

- You have imported the mappings to your local workspace from the On-Premise repository.

  > **Note**
  >
  > For more information about how to import mappings from the ES Repository, see Importing SAP NetWeaver PI Objects from On-Premise Repository [page 16].

Context

You perform this task to assign a mapping that is available in your local workspace.Your local workspace can contain mappings that are already imported from an external repository, such as the ES Repository, or you have obtained them from some other integration project.

> **Caution**
>
> You have to be cautious in distributing the mappings imported from ES Repository if they contain any sensitive data. Although you securely import mappings from the repository by providing the user credentials, copying the imported mappings from one Integration Project to another does not require any authorization.

In an integration flow project, the *src.main.resources.mapping* package can contain message mapping (.mmap), operation mapping (.opmap), XSL mapping and XSLT mapping.

Procedure

1. To add a mapping element to an integration flow model, perform the steps below:
   a. In the *Model Configuration* editor, select the sequence flow within the pool.
   b. From the context menu of the sequence flow, choose *Add Mapping*.
2. Select the mapping in the integration flow and open the *Properties* view.
3. In the *Mapping* tab, choose *Browse...* to select a mapping for the integration flow.

   > **Note**
   >
   > You can assign an operation mapping only if the sender and receiver channel is configured with SOAP (SAP RM) or SOAP (SOAP 1.x) connectors.

4. In the *Choose Mapping* dialog, choose the dropdown icon and select *Local* to obtain the mapping.

   > **Note**
   >
   > **Local** option shows mapping of different types, such as message mapping (.mmap), operation mapping (.opmap), XSL mapping and XSLT mapping, available within the current project.

5. Select a mapping.
6. Choose *OK*.

   > **Note**
   >
   > To view or modify the definition of message mapping in the mapping editor, click *Name* in the *Properties* view. If you have assigned an operation mapping, the navigation to the mapping editor is not supported.

   > **Restriction**
   >
   > Message mapping does not work if the integration flow project contains whitespaces.

2.2.6.3.2 Defining Content Modifier

You use this task if you want to modify the content of the incoming message by providing additional information in the header or body of a message before sending it to the receiver.

Context

You can also use content modifier to define local properties for storing additional data during message processing. These properties can be further used in connectors and conditions.

Example

In SFTP Connector, the user can specify the receiver file name as - File_${property.count}.txt.

In Gateway, the user can define the condition as ${property.count} = '5'.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. If you want to add a content modifier in the integration flow, choose *Add Message Transformers* > *Content Modifier* from the context menu of a connection within the pool.
3. Select the added content modifier in the integration flow model to configure it.
4. To configure the content modifier with a saved configuration that is available as a template, choose *Load Element Template* from the context menu of the *Content Modifier* element.
5. In the *Properties* view, select the **Header** tab.
6. In the new row, define additional headers or set values for existing headers of messages by using constants, another header, an XPath, a property, an external parameter, or by forming an expression.
   a. To enter an XPath, select *xpath* in the *Type* column and browse for an XPath from the lookup in the *Value* column.

      > **Note**
      > - The *Data Type* column is mainly used for the XPath type. The data type can belong to any Java class. An example of a data type for an XPath is *java.lang.String*.
      > - If the XPath contains a namespace prefix, specify the association between the namespace and the prefix on the *Runtime Configuration* tab page of the integration flow *Properties* view.

   b. To enter a header, select *header* in the *Type* column, and browse for a header from the lookup in the *Value* column.

      > **Note**
      > - This lookup dialog shows you a list of headers that you have specified:
      >   - In the *Header* tab page of *Content Modifier* flow steps
      >   - In the *Allowed Headers* field of the *Runtime Configuration* tab page

   c. To enter an external parameter, select *external parameter* in the *Type* column and define the parameter key. For more information, see **Externalizing Parameters of Integration Flow** (section 2.5).
7. In the same way as for headers, the user can also define properties with different value types as explained above, by selecting the **Property** tab in the *Properties* view.

   > **Note**
   > - Header values can be lost following the external system call, whereas properties will be available for complete message execution.
   > - During outbound communication, headers will be handed over to all message receivers and integration flow steps, whereas properties will remain within the integration flow and will not be handed over to receivers.
8. Save the changes.
9. To save the configuration of the modifier as a template, choose *Save as Template* from the context menu of the *Content Modifier* element.

> **Note**
>
> When you save the configuration of the *Content Modifier* element as a template, the tool stores the template in the workspace as *<ElementTemplateName>.fst*.

Example

Suppose the incoming message has the following information:

```
<order>
    <book>
        <BookID>A1000</BookID>
<Count>5<Count>
</book>
</order>
```

In the *Body* tab of the *Content Modifier*, you specify the content expected in the outgoing message. Keep a placeholder for the header information to modify the content as shown below:

```
<invoice>
<vendor>${header.vendor}</vendor>
${in.body}
<deliverydate>${header.date}</delivery>
</invoice>
```

In the Header tab of the *Content Modifier*, enter the following:

Table 16:

Name	Type	Value
vendor	constant	ABC Corp
delivery date	constant	25062013

The output message would look like this:

```
<invoice>
<vendor>ABC Corp</vendor>
    <order>
    <book>
        <BookID>A1000</BookID>
<Count>5<Count>
</book>
</order>
<deliverydate>25062013</deliverydate>
</invoice>
```

Related Information

Specifying an Application ID [page 71]

2.2.6.3.2.1 Specifying an Application ID

With the Content Modifier you can specify an application ID that can be used for end-to-end tracing.

To specify an application ID, open the corresponding integration flow and in the Content Modifier step define an **`SAP_ApplicationID`** header element.

To do that, in tab *Header* add an entry with the following attribute-value combination:

Table 17: Content Modifier Configuration

Attribute	Value
Name	**`SAP_ApplicationID`**
Type	Select the following value: *xpath*
Value	As *Type*, select the XPath expression that points to the message element which is to be used as application ID.

When you monitor the messages at runtime, you can then search for all messages whose such defined application ID have a specific value (shows up as *MessageID* attribute in the Message Monitoring editor).

2.2.6.3.3 Defining Encoders

You use this task to encode messages using an encoding scheme to secure any sensitive message content during transfer over the network.

Context

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. If you did not select the encoder pattern when creating the integration flow, choose *Add Message Transformers* > *Encoder* from the context menu of a connection within the pool.
3. Select the encoder in the integration flow model to configure it.
4. In the *Properties* view, select the encoding scheme from the dropdown list. You can select one of the following encoding schemes:
   - Base64 Encode
     Allows you to encode the message content using base64.
   - GZIP Compress
     Allows you to compress the message content using GNU zip (GZIP).
   - ZIP Compress

Allows you to compress the message content using zip.

5. Save the changes.

Example

Consider the input XML payload structure to the encoder:

```
<message>
    Input for encoder
</message>
```

If you select Base64 Encode, the output message would look like this:

```
PG1lc3NhZ2U+DQoJSW5wdXQgZm9yIGVuY29kZXINCjwvbWVzc2FnZT4NCg==
```

2.2.6.3.4 Defining Decoders

You use this task to decode the message received over the network to retrieve original data.

Context

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. If you want to add a decoder in the integration flow, choose *Add Message Transformers* > *Decoder* from the context menu of a connection within the pool.
3. Select the newly added decoder in the integration flow model to configure it.
4. In the *Properties* view, select the decoding scheme from the dropdown list. You can select one of the following decoding schemes:
   - Base64 Decode
     Allows you to decode base64-encoded message content.
   - GZIP Decompress
     Allows you to decompress the message content using GNU zip (GZIP).
   - ZIP Decompress
     Allows you to decompress the message content using zip.
5. Save the changes.

Example

Let us suppose that an input message to the decoder is a message encoded in Base64 that looks like this:

```
PG1lc3NhZ2U+DQoJSW5wdXQgZm9yIGVuY29kZXINCjwvbWVzc2FnZT4NCg==
```

The output message of the decoder would be as follows:

```
<message>
    Input for encoder
</message>
```

2.2.6.3.5 Defining Content Filter

Context

You use this task if you want to filter information by extracting a specific node from the incoming message.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor
2. If you want to add content filter in the integration flow, choose *Add Message Transformers* > *Content Filter* from the context menu of a connection within the pool.
3. Select the added content filter in the integration flow model to configure it.
4. In the *Properties* view, enter an Xpath to extract a specific message part from the body. For example, in the *Xpath* field, enter **`/ns0:MessageBulk/Message/MessageContent/text()`** .

   > **Tip**
   >
   > To quickly specify the Xpath in the content filter, select *Define Content Filter* from the context menu of the element.

5. Save the changes.

> **Example**
>
> Consider an XML payload structure-

```
<Message>
<orders>
        <order>
            <clientId>I0001</clientId>
            <count>100</count>
        </order>
        <order>
            <clientId>I0002</clientId>
                  <count>10</count>
        </order>
</orders>
</Message>
```

If you enter an Xpath- `/Message/orders/order/count/text()`. The output of the Content Filter would be- `10010`

2.2.6.3.6 Defining Message Digest

This integration flow step is used to calculate a digest of the Payload or parts of it and store the result in a message header.

Context

In detail, the *Message Digest* integration flow step transforms a message into a canonical XML document. From this document, a digest (hash value) is calculated and written to the message header.

> **Note**
>
> Canonicalization transforms an XML document into a form (the canonic form) that makes it possible to compare it with other XML documents. With the Message Digest integration flow step, you can apply canonicalization to a message (or to parts of a message), calculate a digest out of the transformed message, and add the digest to the message header.
>
> In simple terms, canonicalization skips non-significant elements from an XML document. To give some examples, the following changes are applied during the canonicalization of an XML document: Unification of quotation marks and blanks, or encoding of empty elements as start/end pairs.

The original message remains unchanged.

You also have the option to define a filter to apply canonicalization only to a sub-set of the message.

Procedure

1. Open the integration flow in the *Model Configuration* editor.
2. Select the *Message Digest* step and add it to the integration flow.
3. Specify the following attributes for the step.

Option	Description
Filter	If you only want to transform part of the message, enter an XPath expression to specify the part (optional attribute). You can also define a prefix namespace mapping (open the *Properties* view to do that).
Canonicalization Method	You can choose between the following methods. ○ Inclusive XML Canonicalization More information: http://www.w3.org/TR/2001/REC-xml-c14n-20010315

Option	Description
	○ Inclusive XML Canonicalization with Comments More information: http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments ○ Exclusive XML Canonicalization More information: http://www.w3.org/2001/10/xml-exc-c14n ○ Exclusive XML Canonicalization with Comments More information: http://www.w3.org/2001/10/xml-exc-c14n#WithComments This is an optional attribute.
Digest Algorithm	Select the hash algorithm to be used to calculate the digest. You can choose between the following hash algorithms: ○ SHA-1 ○ SHA256 ○ SHA384 ○ SHA512 ○ MD5 This is a mandatory attribute.
Header Name	Enter the name of the header element which is to contain the hash value (digest). This is a mandatory attribute.

2.2.6.3.7 Defining Converter

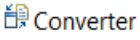
Context

The converter element enables you to transform an input message in one format to another.You can also specify some conditions that are used as reference to transform the message.

The CSV to XML converter transforms a message in CSV format to XML format. Conversely, the XML to CSV converter transforms a message in XML format to CSV format.

Procedure

You use this procedure to define a converter element in an integration process.

1. Open the integration flow in the *Model Configuration* editor.
2. Access the palette and choose *Message Transformers*.
3. Drag  from palette to the integration process.

> **Note**
>
> Alternatively, you can also add the converter from context menu. Choose *Message Transformers* > *Converter*

4. Make the necessary connections to add *Converter* as a part of the integration flow.
5. If you want to use *CSV to XML Converter*, perform the following substeps.
   a. Select the *Converter* element and access *Properties* tab page.
   b. Provide values in fields based on description in table.

Field	Description
XML Schema File Path	Choose *Browse* and select the file path to the XML schema file that is used as the basis for message transformation. The XML file schema format is used as the basis for creation of the payload. This file has to be in the package source.main.resources.xsd
Path to Target Element in XSD	XPath in the XML Schema File where the content from CSV has to be placed.
Record Identifier in CSV	The corresponding record in the CSV file that has to be considered for conversion. This entry is the first text in every new record of the CSV. **Note** If this value is not specified then all the records would be considered for conversion.

   c. In the *Field Separator in CSV* field, choose the character that you want to use as the field separator in CSV file from the dropdown list.

> **Note**
>
> If you want to use a field separator that is not available in the dropdown list, manually enter the character in *Field Separator in CSV* field.

   d. Save changes.

6. If you want to use *XML to CSV Converter*, perform the following substeps.
   a. In the context menu of *Converter*, choose *Switch to XML to CSV Converter*.
   b. Access the *Properties* tab page.
   c. In the *Path to Source Element* field, enter the path of the source element in XML file.
   d. In the *Field Separator in CSV* field, select the field separator from the dropdown list.
   e. If you want to use the field names in the XML file as headers in CSV file, select the *Field Names as Headers in CSV* checkbox.
   f. If you want to include the parent element of the XML file in the CSV file, in the *Advanced* tab page select *Include Parent Element* checkbox.

g. If you want to include the attribute values of the XML file in the CSV file, in the *Advanced* tab page select *Include Attribute Values* checkbox.
h. Save changes.

Example

Consider a CSV file that is in the following format:

```
COMPETENCY,
Role Specific,
Computer Skills,
"Skilled in the use of computers, adapts to new technology, keeps abreast of
changes, learns new programs quickly, uses computers to improve productivity."
```

Consider the Schema XML File in the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<CompetencyList>
<Competency>
<id></id>
<name></name>
<description></description>
</Competency>
</CompetencyList>
```

Value for the field *XPath of Record Identifier in XSD* is given as **CompetencyList\Competency**.

After it is processed by the CSV to XML converter, the XML output appears in the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<CompetencyList>
<Competency>
<id>Role Specific</id>
<name>Computer Skills</name>
<description>Skilled in the use of computers, adapts to new technology, keeps
abreast of changes, learns new programs quickly, uses computers to improve
productivity.</description>
</Competency>
</CompetencyList>
```

2.2.6.3.8 Defining the XML-to-JSON Converter

The XML-to-JSON Converter enables you to transform messages in XML format to JSON format and messages in JSON format to XML format.

Context

The XML-to-JSON converter transforms a message in XML format to JSON format. Conversely, the JSON-to-XML converter transforms a message in JSON format to XML format.

The conversion from **XML to JSON** format and **JSON to XML** format follows the following rules:

- An element is represented as a JSON member whose name is the concatenation of the JSON prefix corresponding to the XML namespace, JSON delimiter, and the element name. If the element has no namespace, then no prefix or JSON delimiter is added to the JSON name.
- An attribute is represented as a JSON member whose name is the concatenation of '@' , the JSON prefix corresponding to the XML namespace, JSON delimiter, and the attribute name. If the attribute has no namespace, then no prefix or JSON delimiter is added to the JSON name.
- An element with no characters or child elements is represented by "element" : "" .
- An element with multiple child elements of the same name is represented by an array. This also applies if children with another name reside between children with the same name. For example, <root><childA>A1</childA><childB>B</childB><childA>A2</childA></root> is transformed to {"root": {"childA":["A1","A2"],"childB":"B"}}, which means that the order of the children is not preserved.
- The value of a complex element (with attributes, for example) is represented by a separate member "$":"value" .

The conversion from **XML to JSON** format follows the following rules:

- Elements with mixed content (for example. <A>mixed1_value<B>valueB</B>mixed2_value</A>) are not supported. You will get incorrect results for XML to JSON: {"A": {"B":"valueB","$":"mixed1_valuemixed2_value"}}.
- No namespace declaration is written into the JSON document.
- Tabs, spaces, and new lines between elements and attributes are ignored.
- If you have an element with a namespace but without an XML prefix whose namespace is not contained in the XML-Namespace-to-JSON-Prefix map, you get an exception: <A xmlns="http://test"/> -> IllegalStateException Invalid JSON namespace: http://test.

The conversion from **JSON to XML** format follows the following rules:

- The result XML document is encoded in UTF-8 and gets the XML header "<?xml version='1.0' encoding='UTF-8'?>".
- The content of the XML-Namespace-to-JSON-Prefix map is transformed to namespace prefix declarations on the root element.
- A member with a JSON null value is transformed to an empty element, for example, "C":null -> <C/>.
- Conversion of "@attr":null to XML is not supported (you get a NullPointerException).
- If no XML namespace is defined for a JSON prefix, the full member name with the prefix and JSON delimiter is set as the element name: "p:A" -> <p:A>.

Procedure

You use this procedure to define a converter element in an integration process.

1. Open the integration flow in the *Model Configuration* editor.
2. Access the palette and choose *Message Transformers*.
3. Drag *Converter* from the palette to the integration process.
4. From the context menu, choose *Switch to XML to JSON Converter*.
5. On the *XML to JSON Converter* properties tab page, define the parameters to convert the XML data format to JSON data format.

Option	Description
Name	Enter the name of the converter.
XML Namespace (only if the option *Use Namespace Mapping* is selected)	Enter the mapping of the XML namespace to the JSON prefix. The JSON namespace/prefix must begin with a letter and can contain aA-zZ and 0-9.
JSON Prefix Separator (only if the option *Use Namespace Mapping* is selected)	Enter the JSON Prefix Separator to be used to separate the JSON prefix from the local part. The value used must not be used in the JSON prefix or local name. The following characters are allowed: Colon(:), comma(,), dot(.), pipe(|), semicolon(;), and space.
JSON Output Encoding	Enter the JSON Output Encoding. The default value is *from header or property*. If you select *from header or property*, the converter tries to read the encoding from the message header or exchange property `CamelCharsetName`. If there is no value defined, UTF-8 is used.

6. Save the changes.
7. If you want to use the JSON-to-XML Converter, in the context menu of the converter choose *Switch to JSON to XML Converter*.
8. On the *JSON to XML Converter* properties tab page, define the parameters to convert the JSON data format to XML data format.

Option	Description
Name	Enter the name of the converter.
JSON Prefix (only if the option *Use Namespace Mapping* is selected)	Enter the mapping of the JSON prefix to the XML namespace. The JSON namespace/prefix must begin with a letter and can contain aA-zZ and 0-9.
JSON Prefix Separator	Enter the JSON Prefix Separator to be used to separate the JSON prefix from the local part. The value used must not be used in the JSON prefix or local name. The following characters are allowed: Colon(:), comma(,), dot(.), pipe(|), semicolon(;), and space.
JSON Input Encoding	Enter the JSON Input Encoding. The default value is *from header or property*. If you select *from header or property*, the converter tries to read the encoding from the message header or exchange property `CamelCharsetName`. If there is no value defined, UTF-8 is used.

9. Save the changes.

Example

The following example shows the transformation from *XML to JSON* (option *Use Namespace Mapping* is selected). In this example the following character is used as the JSON Prefix Separator: ':'

XML Document

```
<?xml version='1.0' encoding='UTF-8'?>
<n1:A xmlns:n1="http://com.sap/abc" xmlns:n2="http://com.sap/xyz">
    <n2:B n2:attr1="1" attr2="2">valueB</n2:B>
    <C>valueC1</C>
    <C>valueC2</C>
    <D/>
    <E>valueE<E>
</n1:A>
```

With the following XML namespace to JSON prefix mapping

XML Namespace	JSON Prefix
http://com.sap/abc	abc
http://com.sap/xyz	xyz

you get the following JSON document:

JSON Document

```
{"abc:A":{"xyz:B":{"@xyz:attr1":"1","@attr2":"2","$":"valueB"},"C":
["valueC1","valueC2"],"D":"","E":"valueE"}}
```

Note

Line breaks and spaces between the elements are ignored.

The following example shows the transformation from *JSON to XML* (option *Use Namespace Mapping* is selected). In this example the following character is used as the JSON Prefix Separator: ':'

JSON Document

```
{"abc:A":{"xyz:B":{"@xyz:attr1":"1","@attr2":"2","$":"valueB"},"C":
["valueC1","valueC2"],"D":"","E":"valueE"}}
```

With the following JSON prefix to XML namespace mapping

JSON Prefix	XML Namespace
abc	http://com.sap/abc
xyz	http://com.sap/xyz

you get the following XML document:

XML Document

```
<?xml version='1.0' encoding='UTF-8'?>
<abc:A xmlns:abc="http://com.sap/abc" xmlns:xyz="http://com.sap/xyz"><xyz:B
xyz:attr1="1" attr2="2">valueB</xyz:B><C>valueC1</C><C>valueC2</C><D/
><E>valueE<E></abc:A>
```

The following example shows the transformation from *XML to JSON* (option *Use Namespace Mapping* is **not** selected). In this example the following character is used as the JSON Prefix Separator: ':'

XML Document

```
<?xml version='1.0' encoding='UTF-8'?>
<abc:A xmlns:abc="http://com.sap/abc" xmlns:xyz="http://com.sap/xyz"><xyz:B
xyz:attr1="1" attr2="2">valueB</xyz:B><C>valueC1</C><C>valueC2</C><D/
><E>valueE</E></abc:A>
```

is transformed to

JSON Document

```
{"A":{"B":{"@attr1":"1","@attr2":"2","$":"valueB"},"C":
["valueC1","valueC2"],"D":"","E":"valueE"}}
```

The following example shows the transformation from *JSON to XML* (option *Use Namespace Mapping* is **not** selected). In this example the following character is used as the JSON Prefix Separator: ':'

JSON Document

```
{"abc.A":{"xyz.B":{"@xyz.attr1":"1","@attr2":"2","$":"valueB"},"C":
["valueC1","valueC2"],"D":"","E":"valueE"}}
```

is transformed to

XML Document

```
<?xml version='1.0' encoding='UTF-8'?><A><B attr1="1" attr2="2">valueB</
B><C>valueC1</C><C>valueC2</C><D/><E>valueE<E></A>
```

Related Information

Limitations for XML-to-JSON Conversion [page 81]

2.2.6.3.8.1 Limitations for XML-to-JSON Conversion

The following limitations apply to XML-to-JSON and JSON-to-XML conversion:

- The XML element and attribute names must not contain any delimiter character because the delimiter is used in JSON to separate the prefix from the element name.
- Elements with mixed content are not supported.
- Streaming is not supported: For JSON to XML, the JSON document is first parsed completely into a JSON model before it is serialized to XML; for XML to JSON as well, a complete JSON model is built before the JSON model is serialized.

The following limitations apply only to XML-to-JSON conversion:

- XML comments (<!-- comment -->) are not represented in the JSON document, they are ignored.
- DTD declarations are not represented in the JSON document, they are ignored.

- XML processing instructions are not represented in the JSON document, they are ignored.
- No conversion to JSON primitive types for XML to JSON. All XML element/attribute values are transformed to a JSON string.
- Entity references (except the predefined entity references & < > " ') are not represented in the JSON document, they are ignored.
- If a sibling with another name resides between XML sibling nodes with the same name, then the order of the siblings is not kept in JSON, because siblings with the same name are represented by one array. Example: <root><childA>A1</childA><childB>B</childB><childA>A2</childA></root> -> {"root":{"childA":["A1","A2"],"childB":"B"}}
- If you have an element with namespace but without XML prefix whose namespace is not contained in the XML-namespace-to-Json-prefix map then you get an exception: <A xmlns="http://test"/> -> IllegalStateException Invalid JSON namespace: http://test

The following limitations apply only to JSON-to-XML conversion:

- Conversion of "@attr":null to XML is not supported (you get a NullPointerException).
- JSON member names must not contain in their local or prefix part any characters that are not allowed for XML element/attribute names (for example space or column (':') are not allowed). XML element/attribute names are of type NCName (see http://www.w3.org/TR/1999/REC-xml-names-19990114/#NT-NCName).
- JSON texts that starts with an array (for example, [{"a":{"b":"bvalue"}}]) are not supported. The JSON text must start with a JSON object like {{"a":{"b":"bvalue"}}}. (javax.xml.stream.XMLStreamException is thrown.)
- The root JSON object must contain exactly one member. If the root JSON object contains more than one member (for example, {"a":"avalue","b":"bvalue"}), then a javax.xml.stream.XMLStreamException is thrown. If the root JSON object is empty ({}), then a javax.xml.stream.XMLStreamException is thrown.
- If for a JSON prefix no XML Namespace is defined then the full member name with the prefix and JSON delimiter is set as element name: "p:A" -> <p:A>

2.2.6.3.9 Defining Script

Context

You use this task to execute custom java script or groovy script for message processing.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. Add script in the integration flow, choose *Add Message Transformers*>*Script* from the context menu of a connection within the pool.
3. Select the added script in the integration flow model to configure it.

4. If you already have an existing java script or groovy script, copy it to the folder src.main.resources.script. To assign this script, choose *Assign Script* from the context menu of the *Script* element.
5. To create a new java script or groovy script with the file extension .gsh, choose *New Script* from the context menu of the *Script* element.
6. Specify a custom function that will take the “message” object as the argument in *Script Function*.

   In the script you require *Script Function*, which will be executed at runtime. The function definition is as follows:

```
import com.sap.gateway.ip.core.customdev.util.Message;
def Message processData(Message message) {
    def body = message.getBody();
    //modify body
        message.setBody(body + "enhancements");

        //modify headers
        def map = message.getHeaders();
        def value = map.get("oldHeader");
        message.setHeader("oldHeader", value + "modified");
        message.setHeader("newHeader", "headerValue");

       //modify properties
        map = message.getProperties();
        value = map.get("oldProperty");
        message.setProperty("oldProperty", value + "modified");
        message.setProperty("newProperty", "script");

                return message;
}
```

7. Add or modify Header, Body, and Property by using the interfaces below on the "message" object.
   1. You can use the following interfaces for Header:
      - public java.util.Map<java.lang.String,java.lang.Object> getHeaders()
      - public void setHeaders(java.util.Map<java.lang.String,java.lang.Object> exchangeHeaders)
      - public void setHeader(java.lang.String name, java.lang.Object value)
   2. You can use the following interfaces for Body:
      - public java.lang.Object getBody()
      - public void setBody(java.lang.Object exchangeBody)
   3. You can use the following interfaces for Property:
      - public java.util.Map<java.lang.String,java.lang.Object> getProperties()
      - public void setProperties(java.util.Map<java.lang.String,java.lang.Object> exchangeProperties)
      - public void setProperty(java.lang.String name, java.lang.Object value)

   > **Note**
   - You should not add or modify a Property name starting with sap.
   - If no Script Function is specified in Integration Flow, the Script Function name will be processData by default.

8. Save the changes.

   > **Note**
   - The script should be saved in the src.main.resources.script folder.

- In the *Properties* view, you can also choose *Browse* to select java scripts or groovy scripts for custom processing.
- You can also store external jar(s) in *src.main.resources.lib*. You can then invoke functions from these external jar(s) in the script.

2.2.6.4 Defining Message Persistence

2.2.6.4.1 Defining Data Store

You can use data store to store messages. Data stores support four types of operations.

Context

Data store supports the following operations:

- Write – You can use this operation to store the messages in the data store.
- Delete – You can use this operation to trigger the deletion of messages in the data store.
- Select – You can use this operation to fetch messages in bulk from the data store. You can also specify the number of messages you fetch in each poll.
- Get – You can use this operation to fetch a specific message from the data store.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. If you want to add *Data Store* in the integration flow, choose *Add Message Persistence*->*Data Store Operations* from the context menu of a connection within the pool.

   > **Note**
   >
   > Adding data store operations enables *Write* operations by default. The *Switch to* option allows you to choose a different operation.

3. If you use a *Write* operation, you can store the messages in the data store by configuring the data store name and a unique *Entry ID*.
4. Configure the behavior of stored messages. For example, specify the time by which the messages have to be fetched before an alert is raised, or the number of days after which the stored messages are deleted.
5. You can also switch to *Delete*, *Select*, and *Get* operations.

   > **Note**
   >
   > - For *Write*, *Delete*, and *Select* operations, define the *Entry ID*.

- In the case of write operations, if the *Entry ID* is not defined, the data store component generates an entry ID.
- In the case of delete and get operations, you can explicitly define an *Entry ID* or pass header SapDataStoreId.

- In the case of *Select* operations, you can select multiple messages. The content of the selected messages will appear in the following format:
  <messages>
  <message id="<Entry ID>">
  <Content of selected message>
  </message>
  <message id="<Entry ID>">
  .....
  .....
  .....
  </message>
  <message id="<Entry ID>">
  .....
  .....
  .....
  </message>
  </messages>
- The minimum value of *Expiration Period* should be at least twice that of *Retention Threshold for Alerting*.

6. Save the changes.

> **Note**
>
> The default data store name for variables is 'sap_global_store'. You should not use this value as the data store name for data store operations.

2.2.6.4.2 Defining Write Variables

You use write variables to specify values for variables and support message flow execution. You can use these variables across message flows for a specific integration flow or across integration flows.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. To add write variables in the integration flow, choose *Add Message Persistence* > *Write Variables* from the context menu of a connection within the pool.
3. Select the *Write Variables* tab in the *Properties* view.
4. Choose *Add*.
5. Specify a name for the variable, and perform the following substeps to assign values to the variable.

a. To assign a value using an XPath, select *xpath* in the *Type* column and enter the *XPath* expression in the *Value* column.

.

> **Note**
>
> - The *Data Type* column is applicable for the xpath and expression types. The data type can be any Java class. An example of a data type for an XPath is java.lang.String.
> - If the XPath contains a namespace prefix, specify the association between the namespace and the prefix on the *Runtime Configuration* tab page of the integration flow Properties view.

b. To assign a value using a header, select header in the *Type* column and enter the header in the *Value* column.

c. To assign an external parameter, select external parameter in the *Type* column and define a parameter key. For more information, see Externalizing Parameters of Integration Flow.

d. To assign a property, select property in the *Type* column. Properties are local variables. For more information about properties, see Defining Content Modifier [page 68] .

6. Save the changes.

> **Note**
>
> - If you want the variable to be used in multiple integration flows, select the global scope checkbox.
> - By default, stored variables are deleted 90 days after the last update; the system raises an alert 2 days before the variables expire.
> - The default data store name for variables is ‘sap_global_store’. You should not use this value as the data store name for data store operations.

2.2.6.5 Defining Message Routing

2.2.6.5.1 Defining Gateway

Prerequisites

You have selected the *Content-Based Router (Multiple Interfaces/Multiple Receivers)* pattern or you have added the router element to the integration flow model from the palette.

Context

You perform this task when you have to specify conditions based on which the messages are routed to a receiver or an interface during runtime. If the message contains XML payload, you form expressions using the

Xpath-supported operators. If the message contains non-XML payload, you form expressions using the operators shown in the table below:

Table 18: Usage of Operators in Non- XML Conditions

Operator	Example
=	${header.SenderId} = '1'
!=	${header.SenderId} != '1'
>	${header.SenderId} > '1'
>=	${header.SenderId} >= '1'
<	${header.SenderId} < '1'
<=	${header.SenderId} <= '1'
and	${header.SenderId}= '1' and ${header.ReceiverId} = '2'
or	${header.SenderId}= '1' or ${header.ReceiverId}= '2'
contains	${header.SenderId} contains '1'
not contains	${header.SenderId} not contains '1'
in	${header.SenderId} in '1,2'
not in	${header.SenderId} not in '1,2'
regex	${header.SenderId} regex '1.*'
not regex	${header.SenderId} not regex '1.*'

Example

A condition with expression `${header.SenderId} regex '1.*'` routes all the messages having Sender ID starting with 1'.

Note

You can define a condition based on an exception that may occur. If condition `${exception.message}contains 'java.lang.Exception'` is true then gateway routes the message to a particular receiver else it routes to other receiver.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. If you want to add a gateway at any point in the integration flow, follow the steps below:
   a. Right-click a connection within the pool and choose *Add Routing*.
   b. Right-click the added *Gateway* notation and choose the relevant option, either *Add Receiver* or *Add Interface*.

3. In the *Model Configuration* editor, select one of the routing branches splitting from the gateway.
4. On the *Route* tab of the *Properties* view, enter a descriptive name for the routing branch.
5. From the *Expression Type* dropdown, select the type of expression used to formulate the routing condition.
   a. Select *XML* to form an expression using Xpath.
   b. Select *Non-XML* to form an expression using message header and message body.
6. In the *Condition* field, formulate a valid XML or non-XML condition that routes the message to its associated receiver.

> **Recommendation**
>
> We recommend you to ensure that the routing branches of a gateway are configured with the same type of condition, either XML or non-XML, and not a combination of both. At runtime, the specified conditions are executed in the same order as displayed in the *Properties* view of the *Gateway*. If the conditions are a combination of both non-XML and XML type, the evaluation fails.

> **Tip**
>
> To quickly configure routing conditions, select *Define Condition* from the context menu of the routing branch.

7. If you want to set the selected route as the default, such that, its associated receiver handles the error situation if no receiver is found, select the *Default Route* option.

> **Note**
>
> Only the route that was recently selected as *Default Route* is considered as the default route.

> **Tip**
>
> To quickly set a route as a default route from the Model Configuration editor tab page, select *Set as Default Route* from the context menu of routing branch.

8. In the *Model Configuration* editor, select the *Gateway* notation.
9. In the *Properties* view, under *Routing Behavior*, select *Raise Alert* or *Throw Exception* to set the behavior if no receiver is found.

> **Note**
>
> To configure an alert, see the Alert Configuration section in the Operations Guide.

10. To model the integration flow when no receiver is found, follow the steps below:
    a. From the *Palette* select the *Terminate End* notation and drop it inside the pool in the *Model Configuration*.
    b. Place the cursor on the *Gateway* to select the connection from the hover menu and drag the connection to the *Terminate End*notation.

> **Note**
>
> If you do not add the *Terminate End* notation in your integration flow model, the *Raise Alert* or *Throw Exception* option does not apply.

11. Select the *Gateway* notation and, in the *Properties* view, check the overview of all the routes and their conditions that you have defined for the scenario.

2.2.6.5.2 Defining Splitter

Prerequisites

You have selected the *Splitter* pattern or you have explicitly added the splitter element to the integration flow model.

Context

You perform this task if you need to break down a composite message into a series of individual messages and send them to a receiver. You split the composite message based on the message type, for example, IDoc or PKCS#7/CMS Signature-Content, or the manner in which the message is to be split, for example, general or iterative splitter.

Example

The following XML payload structure is input into the *Splitter*:

```
<orders>
    <order>
        <clientId>I0001</clientId>
        <count>100</count>
    </order>
    <order>
        <clientId>I0002</clientId>
        <count>10</count>
    </order>
</orders>
```

If we use the XML node *order* as a token, the XML payload is split into two output files.

Output1.xml contains:

```
<order>
    <clientId>I0001</clientId>
    <count>100</count>
</order>
```

Output2.xml contains:

```
<order>
    <clientId>I0002</clientId>
```

```
    <count>10</count>
</order>
```

There is also the *FSN Splitter* option, which splits an FSN bulk message into individual FSN messages and transforms these FSN messages into the internal message format.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. If you want to add a splitter in the integration flow, from the context menu of the connection within the pool, choose *Add Splitter*.
3. In the *Model Configuration* editor, select the splitter element.
4. In the *Properties* view, select a splitter type.

   > **Note**
   >
   > - *IDoc Splitter* is used for composite IDoc messages to be split into a series of individual IDoc messages with the enveloping elements of the composite IDoc message.
   > - *PKCS#7/CMS Signature-Content Splitter* is used when an agent sends a PKCS7 Signed Data message that contains signature and content. This splitter type breaks down the signature and content into separate files.
   > - *General Splitter* splits a composite message comprising of N messages into N individual messages each containing one message with the enveloping elements of the composite message.
   > - *Iterative Splitter* splits a composite message into a series of messages without copying the enveloping elements of the composite message.

5. Configure the details of the selected splitter type.

   > **Tip**
   >
   > For quick information about the fields, check the tooltips.

6. Save the changes.

2.2.6.5.3 Defining an Aggregator

You can use the Aggregator step to combine multiple incoming messages into a single message.

Context

> **Note**
>
> When you use the Aggregator step in combination with a polling SFTP sender adapter and you expect a high message load, please consider the following recommendation:
>
> For the involved in SFTP sender adapter set the value for *Maximum Messages per Poll* to a small number which is larger than 0 (for example, **`20`**) (under *Advanced Parameters*). That way, you ensure a proper message processing status logging at runtime.

Procedure

1. Open the related integration flow and select an *Aggregator* step (in the palette under *Message Routing*).
2. Open the *Properties* view for the step.
3. Specify which incoming messages belong together and are to be aggregated. To do that, open *Correlation* tab and enter an XPath expression that identifies an element based on which the incoming messages should be correlated (in the field *Correlation Expression (XPath)*).

   This attribute determines the messages which are to be aggregated in the following way: messages with the same value for the message element defined by the XPATH expression are being stored in the same aggregate.
4. Open the *Aggregation Strategy* tab and specify the following attributes:

Option	Description
Incoming Format	The content type of the incoming message. Currently, only **`XML (Same Format)`** can be selected. It is important that the incoming messages have the same format.
Aggregation Algorithm	Specifies how the correlated messages should be treated. You can select one of the following options: ○ *Combine in Sequence* Aggregated message contains all correlated incoming messages, and the original messages are put into a sequence. ○ *Combine* Aggregated message contains all correlated incoming messages.

Option	Description
	The *Aggregator* step generates an SAP Process Integration **multi mapping** as specific format. This is important to know because the integration developer typically has to create a mapping from the generated format to the message structure required by the receiver.
Message Sequence Expression (XPath) **(only if for *Aggregation Algorithm* the option *Combine in Sequence* has been selected)**	Enter an XPATH expression for that message element based on which a sequence is being defined. You can use only numbers to define a sequence. Each sequence starts with **`1`**.
Last Message Condition (XPath)	Define the condition (`XPATH = value`) to identify the last message of an aggregate. You have the option to use an *Exclusive Gateway* step after the *Aggregator* in order to evaluate the `Camel ${header.CamelAggregatedCompletedBy}` attribute, and, based on the value, decide how to continue. Note that the header attribute can only have one of the following values: ○ **`timeout`** Processing of the aggregate has been finished because the configured *Completion Timeout* has been reached. ○ **`predicate`** Processing of the aggregate has been finished because the *Completion Condition* has been fulfilled.
Completion Timeout	Defines the maximum time between two messages before aggregation is being stopped (*period of inactivity*). You have the option to use an *Exclusive Gateway* step after the *Aggregator* in order to evaluate the `Camel ${header.CamelAggregatedCompletedBy}` attribute, and, based on the value, decide how to continue. Note that the header attribute can only have one of the following values: ○ **`timeout`** Processing of the aggregate has been finished because the configured *Completion Timeout* has been reached. ○ **`predicate`** Processing of the aggregate has been finished because the *Completion Condition* has been fulfilled.
Data Store Name	Enter the name of the transient data store where the aggregated message is to be stored. Note that only **local** data stores (that apply only to the integration flow) can be used. Global data stores cannot be used for this purpose. By default, the following name is provided: `DS-Aggregator-01`.

Option	Description
	Note The Integration Operations feature provides a Data Store Viewer that allows you to monitor your transient data stores.

Results

The Aggregator step creates a message in the multi-mapping format with just one message instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<sm:Messages xmlns:sm="http://sap.com/xi/XI/SplitAndMerge">
<Message1>
...
</Message1>
</sm:Messages>
```

Related Information

Multi-Mappings [page 10]

2.2.6.5.4 Defining Content Enricher

Prerequisites

You have created an integration project and integration flow.

Context

The content enricher adds the content of a payload with the original message in the course of an integration process. This converts the two separate messages into a single enhanced payload. This feature enables you to make external calls during the course of an integration process to obtain additional data, if any.

Consider the first message in the integration flow as the original message and the message obtained by making an external call during the integration process as the lookup message. You can choose between two strategies to enrich these two payloads as a single message:

- Combine
- Enrich

Consider the following original and lookup messages.

Original Message

```
<?xml version='1.0' encoding='UTF-8'?>
<Employees>
	<Employee>
		<EmployeeID>1</EmployeeID>
		<Address>507 - 20th Ave Apt. 2A</Address>
		<HireDate>1992-04-01T00:00:00</HireDate>
		<FirstName>Nancy</FirstName>
		<location>lo_USA</location>
		<LastName>Davolio</LastName>
	</Employee>
	<Employee>
		<EmployeeID>2</EmployeeID>
		<Address>606 5th lane</Address>
		<HireDate>2014-04-01T00:00:00</HireDate>
		<FirstName>John</FirstName>
		<location>lo_AU_SYD</location>
		<LastName>Carter</LastName>
	</Employee>
</Employees>
```

Lookup Message

```
<?xml version='1.0' encoding='UTF-8'?>
<EmpLoyeeLocations>
	<empLocation>
		<locationId>lo_USA</locationId>
		<address_city>San Mateo</address_city>
		<group>NA_WEST</group>
		<name>San Mateo</name>
	</empLocation>
	<empLocation>
		<locationId>lo_AU_SYD</locationId>
		<address_city>Sydney</address_city>
		<group>APAC</group>
		<name>Sydney</name>
	</empLocation>
</EmpLoyeeLocations>
```

If you use *Combine* as the aggregation strategy, the enriched message appears in the following format.

Enriched Message

```
<?xml version="1.0" encoding="UTF-8"?>
<multimap:Messages xmlns:multimap="http://sap.com/xi/XI/SplitAndMerge">
	<multimap:Message1>
		<Employees>
			<Employee>
				<EmployeeID>1</EmployeeID>
				<Address>507 - 20th Ave Apt. 2A</Address>
				<HireDate>1992-04-01T00:00:00</HireDate>
				<FirstName>Nancy</FirstName>
				<location>lo_USA</location>
				<LastName>Davolio</LastName>
			</Employee>
			<Employee>
				<EmployeeID>2</EmployeeID>
				<Address>606 5th lane</Address>
				<HireDate>2014-04-01T00:00:00</HireDate>
```

```
                        <FirstName>John</FirstName>
                        <location>lo_AU_SYD</location>
                        <LastName>Carter</LastName>
                  </Employee>
            </Employees>
      </multimap:Message1>
      <multimap:Message2>
            <EmpLoyeeLocations>
                  <empLocation>
                        <locationId>lo_USA</locationId>
                        <address_city>San Mateo</address_city>
                        <group>NA_WEST</group>
                        <name>San Mateo</name>
                  </empLocation>
                  <empLocation>
                        <locationId>lo_AU_SYD</locationId>
                        <address_city>Sydney</address_city>
                        <group>APAC</group>
                        <name>Sydney</name>
                  </empLocation>
            </EmpLoyeeLocations>
      </multimap:Message2>
</multimap:Messages>
```

Enrich offers you control on how you can merge the original and lookup message. In this example, we consider the node <location> as the reference to enrich the original message with the lookup message.

Consequently, you specify the following values while configuring the Content Enricher.

Section	Field	User input
Original Message	Path to Node	Employees/Employee
	Key Element	location
Lookup Message	Path to Node	EmployeeLocations\emplocation
	Key Element	locationid

The enriched message will be in the following format.

```
<?xml version='1.0' encoding='UTF-8'?>
<Employees>
      <Employee>
            <EmployeeID>1</EmployeeID>
            <Address>507 - 20th Ave Apt. 2A</Address>
            <HireDate>1992-04-01T00:00:00</HireDate>
            <FirstName>Nancy</FirstName>
            <location>lo_USA</location>
            <empLocation>
                  <locationId>lo_USA</locationId>
                  <address_city>San Mateo</address_city>
                  <group>NA_WEST</group>
                  <name>San Mateo</name>
            </empLocation>
            <LastName>Davolio</LastName>
      </Employee>
      <Employee>
            <EmployeeID>2</EmployeeID>
            <Address>606 5th lane</Address>
            <HireDate>2014-04-01T00:00:00</HireDate>
            <FirstName>John</FirstName>
            <location>lo_AU_SYD</location>
            <empLocation>
                  <locationId>lo_USA</locationId>
                  <address_city>San Mateo</address_city>
```

```
                    <group>NA_WEST</group>
                    <name>San Mateo</name>
                </empLocation>
                <LastName>Carter</LastName>
        </Employee>
</Employees>
```

In the enriched message, you can see the content of the lookup message after the node **<location>**.

Procedure

You use this procedure to add *Content Enricher* element to an integration flow.

1. Open the integration flow in *Model Configuration* editor.
2. From the context menu of integration flow, choose *Add Tasks* > *Service Call*.
3. From the context menu of the service call element, choose *Switch to Content Enricher*.
4. In the *Properties* view, choose the *Aggregation Strategy* field.
5. Perform the required subprocedure below based on the strategy you want to use.

Using Combine Strategy

Procedure

You use this subprocedure for using the *Combine* strategy.

1. In the dropdown list, choose *Combine*.
2. Save changes.

Using Enrich Strategy

Procedure

You use this subprocedure for using the *Enrich* strategy.

1. In the dropdown list, choose *Enrich*.
2. Provide values in fields based on description in table.

Section	Field	Description
Original Message	Path to Node	Path to the node in the original message where content has to be enriched. Ensure that you provide it in the format <xx>/<yy>/<zz> with <xx> being the root node and <zz> being the node where new content will reside
	Key Element	Key element in the original message contained in the Path to Node specified above. Ensure that you provide only the key element name.
Lookup Message	Path to Node	Path to the node that would be used in the lookup message to enrich the content. Ensure that you provide it in the format <xx>/<yy>/<zz> with <xx> being the root node and <zz> being the reference node
	Key Element	Key element in the lookup message contained in the Path to Node specified above. Ensure that you provide only the key element name.

3. Save the changes.

2.2.6.5.5 Defining Multicast of Messages

Prerequisites

- You have created an integration project and integration flow.
- You have opened the *Integration Designer* perspective.

Context

Multicast enables you to send the same message to more than one receiver. This allows you to perform multiple operations on the same message in a single integration flow. Without multicast, you require separate integration processes to perform different operations on the same incoming message.

There are two types of multicast that you can choose:

1. Parallel multicast
2. Sequential multicast

Parallel multicast initiates message transfer to all the receiver nodes in parallel.

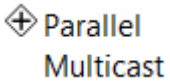
Sequential multicast provides an option to define the sequence in which the message transfer is initiated to the receiver nodes.

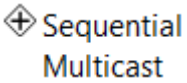
If you want to aggregate the message that is sent to more than one receiver node, you can use the join and gather elements.

> ⚠ Restriction
>
> You cannot use splitter element within a multicast branch

Procedure

You use this procedure to add a multicast and join elements to your integration flow.

1. In the *Model Configuration* editor, access the palette.
2. If you want to add a multicast or join element, perform the following substeps.
   a. Choose *Message Routing*.
   b. If you want to add parallel multicast, drag the 

element from palette to the integration flow.
   c. If you want to add sequential multicast, drag the 

element from the palette to the integration flow.
3. If you want to combine the messages, see Defining Join and Gather [page 98].
4. Add the necessary elements based on your scenario to complete the integration flow.
5. Save the changes.

2.2.6.5.6 Defining Join and Gather

Context

The *Join* element enables you to bring together the messages from different routes before combining them into a single message. You use this in combination with the *Gather* element. *Join* only brings together the messages from different routes without affecting the content of messages.

The *Gather* step enables you to merge messages from more than one route in an integration process. You define conditions based on the type of messages that you are gathering using the *Gather* step. You can choose to gather:

1. XML messages of different format
2. XML messages of the same format
3. Plain text messages

Based on this, you choose the strategy to combine the two messages.

- For XML messages of the same format, you can combine without any conditions (multimapping format) or specify the XPath to the node at which the messages have to be combined.
- For XML messages of different formats, you can only combine the messages
- For plain text messages, you can only specify concatenation as the combine strategy

> **Note**
>
> You need to use the *Join* element before using the Gather element in the integration process.

Procedure

You use this procedure to combine messages using *Gather* in an integration process.

1. Open the integration flow in *Model Configuration* editor.
2. In the palette of the *Model Configuration* editor, choose *Message Routing*.
3. To add *Join*, drag  from the palette to the integration process.
4. Connect all the messages that you want to merge to the  element.
5. To add *Gather*, drag  from the palette to the integration process.
6. In the *Properties* tab page, select value in *Incoming Format* field based on description in table.

Value	Description
XML (Same Format)	If messages from different routes are of the same format
XML (Different Format)	If messages from different routes are of the different format
Plain Text	If messages from different routes are of the plain text format

7. Select value for *Aggregation Algorithm* field based on description given in table.

Incoming Format	Aggregation Algorithm	Description	Additional Fields	Description
XML (Same Format)	Combine	Combine the incoming messages without any conditions. The messages are combined in Multimapping format. **Note** In case you are using the mapping step to map the output of this strategy you can have the source XSD in the LHS and specify the Occurrence as 0..Unbounded.	NA	NA
	Combine at XPath	Combine the incoming messages at the specified XPath	Combine from Source (XPath)	XPath of the node that you are using as reference in the source message to retrieve the information.
			Combine at Target (Path)	Path to node which would act as the root for combined message

Incoming Format	Aggregation Algorithm	Description	Additional Fields	Description
XML (Different Format)	Combine	Combine the incoming messages without any conditions in multi mapping format. **Note** In case you are using the mapping step to map the output of this strategy you can add the XSD's from the different multicast branches one after another in LHS. The sequence of the messages is important and so this strategy makes sense only with the sequential multicast.	NA	NA
Plain Text	Concatenate	Concatenate the information from the different sources one after another	NA	NA

8. Save changes.

2.2.6.6 Defining Message Security

Message-level security features allow you to digitally encrypt/decrypt or sign/verify a message (or both). The following standards and algorithms are supported.

Ensuring security is important, during message exchange, to preserve the integrity of messages. A sender participant uses signatures to ensure authenticity of the sender, and encryption to prevent non-repudiation of messages during the message exchange.

Related Information

Signing the Message Content with PKCS#7/CMS Signer [page 103]
Signing the Message Content with an XML Digital Signature [page 104]
Verifying the PKCS#7/CMS Signature [page 110]
Verifying the XML Digital Signature [page 111]
Encrypting and Signing the Message Content with PKCS#7/CMS [page 114]
Encrypting the Message Content with OpenPGP [page 115]
Decrypting and Verifying the Message Content with PKCS#7/CMS [page 117]
Decrypting the Message Content with OpenPGP [page 118]
Signing the Message Content with Simple Signer [page 102]

2.2.6.6.1 Signing the Message Content with Simple Signer

Simple Signer makes it easy to sign messages to ensure authenticity and data integrity when sending an XML resource to participants on the cloud.

Context

You work with the Simple Signer to make the identity of the sender known to the receiver(s) and thus ensure the authenticity of the messages. This task guarantees the identity of the sender by signing the messages with a private key using a signature algorithm.

In the integration flow model, you configure the Simple Signer by providing a private key alias. The signer uses the alias name to get the private key of type DSA or RSA from the keystore. You also specify the signature algorithm for the key type, which is a combination of digest and encryption algorithms, for example, SHA512/RSA or SHA/DSA. The Simple Signer uses the algorithm to generate the corresponding signature.

Procedure

1. In the *Model Configuration* editor, select the *Signer* element and, from the context menu, select the *Switch to Simple Signer* option.
2. To configure the signer with a saved configuration that is available as a template, choose *Load Element Template* from the context menu of the *Signer* element.
3. On the *Simple Signer* tab page, enter the parameters to create a signature for the incoming message.

Option	Description
Name	Enter a name for the signer. It must consist of alphanumeric ASCII characters or underscores and start with a letter. The minimum length is 3, the maximum length is 30.
Private Key Alias	Enter an alias to select the private key from the keystore.

Option	Description
	You can also specify that the alias is read dynamically from a message header; for example, if you specify ${header.abc} then the alias value is read from the header with the name "abc".
Signature Algorithm	Select a signature algorithm for the RSA or DSA private key type.
Signature Header Name	Enter the name of the message header where the signature value in Base64 format is stored.

4. Save the changes.
5. To save the configuration of the signer as a template, choose *Save as Template* from the context menu of the *Signer* element.

> **Note**
>
> When you save the configuration of the *Signer* element as a template, the tool stores the template in the workspace as <ElementTemplate>.fst.

2.2.6.6.2 Signing the Message Content with PKCS#7/CMS Signer

Context

You work with the PKCS#7/CMS signer to make your identity known to the participants and thus ensure the authenticity of the messages you are sending on the cloud. This task guarantees your identity by signing the messages with one or more private keys using a signature algorithm.

Working with PKCS#7/CMS Signer

In the integration flow model, you configure the PKCS#7/CMS signer by providing one or more private key aliases. The signer uses the alias name to get the private keys of type DSA or RSA from the keystore. You also specify the signature algorithm for each key type, which is a combination of digest and encryption algorithms, for example, SHA512/RSA or SHA/DSA. The PKCS#7/CMS signer uses the algorithm to generate corresponding signatures. The data generated by the signer is known as the Signed Data.

You can choose to include the original content that is to be signed in the Signed Data. This Signed Data is written to the message body. Otherwise, you can choose to include the Signed Data in the `SapCmsSignedData` header and only keep the original content in the message body. You can also Base64-encode the Signed Data in either the message body or the message header, to further protect it during message exchange.

> **Note**
>
> When you Base64-encode the Signed Data, you encode either the message header or body, depending on where the Signed Data is placed. When verifying the message, make sure you specify which part of the message (header or body) was Base64-encoded.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. Right-click a connection within the pool and choose *Add Security Element* > *Message Signer*.
3. In the *Model Configuration* editor, select the *Signer* element.

> **Note**
>
> By default, the *Signer* is of type PKCS#7/CMS. If you want to work with XML Digital Signature, choose the *Switch to XML Digital Signer* option from the context menu.

4. To configure the signer with a saved configuration that is available as a template, choose *Load Element Template* from the context menu of the *Signer* element.
5. In the *Properties* view, enter the details to sign the incoming message with one or more signatures.
6. Save the changes.
7. To save the configurations of the signer as a template, choose *Save as Template* from the context menu of the *Signer* element.

> **Note**
>
> When you save the configuration of the *Signer* element as a template, the tool stores the template in the workspace as <ElementTemplate>.fst.

2.2.6.6.3 Signing the Message Content with an XML Digital Signature

You sign with an XML digital signature to ensure authenticity and data integrity while sending an XML resource to the participants on the cloud.

Context

The following types of XML digital signatures are supported:

- **Enveloping XML Signature**: The input message body is signed and embedded within the signature. This means that the message body is wrapped by the *Object* element, where *Object* is a child element of the *Signature* element.

> **Example**
>
> A template of the enveloping signature is shown below and describes the structure supported by XML signature implementation. ("?" denotes zero or one occurrence; the brackets [] denote variables whose values can vary.)

```
<Signature>
     <SignedInfo>
            <CanonicalizationMethod>
```

```
            <SignatureMethod>
            <Reference URI="#[generated object_id]"
type="[optional_type_value]">
                  <Transform Algorithm="canonicalization method">
                  <DigestMethod>
                  <DigestValue>
            </Reference>
            (<Reference URI="#[generated keyinfo_id]">
                  <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>
                  <DigestMethod>
                  <DigestValue>
            </Reference>)?
     </SignedInfo>
     <SignatureValue>
     (<KeyInfo Id="[generated keyinfo_id]">)?
    <--!The Id attribute is only added if there exists a reference-->
     <Object Id="[generated object_id]"/>
</Signature>
```

- **Enveloped XML Signature**: The digital signature is embedded in the XML message to be signed. This means that the XML message contains the *Signature* element as one of its child elements. The *Signature* element contains information such as:
  - Algorithms to be used to obtain the digest value
  - Reference with empty URI, which means the entire XML resource
  - Optional reference to the KeyInfo element (attribute *Also Sign Key Info*)

Example

A template of the enveloped signature is shown below and describes the structure supported by XML signature implementation. ("?" denotes zero or one occurrence; the brackets [] denote variables whose values can vary):

```
<[parent element]>
      ...
      <Signature>
          <SignedInfo>
                <CanonicalizationMethod>
                <SignatureMethod>
                <Reference URI="" type="[optional_type_value]">
                      <Transform Algorithm="http://www.w3.org/2000/09/
xmldsig#enveloped-signature"/>
                      <Transform Algorithm=[canonicalization method]/>
                      <DigestMethod>
                      <DigestValue>
                </Reference>
                (<Reference URI="#[generated keyinfo_Id]">
                      <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>
                      <DigestMethod>
                      <DigestValue>
                </Reference>)?
          </SignedInfo>
          <SignatureValue>
          (<KeyInfo Id="[generated keyinfo_id]">)?
        <--!The Id attribute is only added if there exists a reference-->
     </Signature>
</[parent element]>
```

- **Detached XML Signature**: The digital signature is a sibling of the signed element. There can be several XML signatures in one XML document.

You can sign several elements of the message body. The elements to be signed must have an attribute of type ID. The ID type of the attribute must be defined in the XML schema that is specified during the configuration.
Additionally, you specify a list of XPath expressions pointing to attributes of type ID. These attributes determine the elements to be signed. For each element, a signature is created as a sibling of the element. The elements are signed with the same private key. Elements with a higher hierarchy level are signed first. This can result in nested signatures.

Example

A template of the detached signature is shown below and describes the structure supported by XML signature implementation. ("?" denotes zero or one occurrence; the brackets [] denote variables whose values can vary):

```
    (<[signed element] Id="[id_value]">
    <!-- signed element must have an attribute of type ID -->
    ...

    </[signed element]>
    <other sibling/>* <!-- between the signed element and the corresponding
signature element, there can be other siblings -->
    <Signature>
    <!-- signature element becomes the last sibling of the signed element -->
        <SignedInfo>
            <CanonicalizationMethod>
            <SignatureMethod>
            <ReferenceURI="#[id_value]"type="[optional_type_value]">
            <!-- reference URI contains the ID attribute value of the signed
element -->
                <TransformAlgorithm=[canonicalization method]/>
                <DigestMethod>
                <DigestValue>
            </Reference>
            (<ReferenceURI="#[generated_keyinfo_Id]">
                <TransformAlgorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315" />
                <DigestMethod>
                <DigestValue>
            </Reference>)?
    </SignedInfo>
    <SignatureValue>
    (<KeyInfoId="[generated_keyinfo_id]">)?

    <!-- The Id attribute is only added if there exists a reference with the
corresponding URI -->
    <Signature>)+
```

Note that the following elements are generated and cannot be configured with the Integration Designer:

- Key Info ID
- Object ID

Working with XML Digital Signer

The sender canonicalizes the XML resource to be signed, based on the specified transform algorithm. Using a digest algorithm on the canonicalized XML resource, a digest value is obtained. This digest value is included within the 'Reference' element of the 'SignedInfo' block. Then, a digest algorithm, as specified in the signature algorithm, is used on the canonicalized SignedInfo. The obtained digest value is encrypted using the sender's private key.

> **Note**
>
> Canonicalization transforms the XML document to a standardized format, for example, canonicalization removes white spaces within tags, uses particular character encoding, sorts namespace references and eliminates redundant ones, removes XML and DOCTYPE declarations, and transforms relative URIs into absolute URIs. The representation of the XML data is used to determine if two XML documents are identical. Even a slight variation in white spaces results in a different digest for an XML document.

Procedure

1. In the *Model Configuration* editor, select the *Signer* element and, from the context menu, select the *Switch to XML Digital Signer* option.
2. To configure the signer with a saved configuration that is available as a template, choose *Load Element Template* from the context menu of the *Signer* element.
3. On the *XML Digital Signer* tab page, enter the parameters to create an XML digital signature for the incoming message.

Option	Description
Private Key Alias	Enter an alias to select the private key from the keystore.
Signature Algorithm	Select a signature algorithm for the RSA or DSA private key type. Using the private key, the sender encrypts the digest.
Digest Algorithm	Select the digest algorithm that is used to calculate a digest from the canonicalized XML document. Note that if the digest algorithm is not specified, the digest algorithm of the signature algorithm is used by default.
Signature Type	○ *Enveloping XML Signature* The input message body is signed and embedded in the signature. This means that the message body is wrapped by the Object element, where Object is a child element of the Signature element. ○ *Enveloped XML Signature* The digital signature is embedded in the XML message to be signed. This means that the XML message contains the Signature element as one of its child elements. ○ *Detached XML Signature* The digital signature is a sibling of the signed element. There can be several XML signatures in one XML document.
XML Schema file path (only if the option **`Detached XML Signatures`** is selected.	Choose *Browse* and select the file path to the XML schema file that is used to validate the incoming XML document. This file has to be in the package source.main.resources.xsd
Signatures for Elements (only if the option **`Detached XML Signatures`** is selected.	Choose *Add* to enter the XPath to the attribute of type ID, in order to identify the element to be signed. Example: /nsx:Document/SubDocument/@Id Namespaces prefixes must be defined in the namespaces mapping of the runtime configuration.

Option	Description
Parent Node (only if the option **`Enveloped XML Signature`** is selected for the attribute *Signature Type*)	Specify how the parent element of the Signature element is to be specified. You have the following options: ○ *Specified by Name and Namespace* (using local name and namespace) ○ *Specified by XPath expression* (using an *XML Path Language* (*XPath*))
Parent Node Name (only if the option **`Enveloped XML Signature`** is selected for the attribute *Signature Type* and **`Specified by Name and Namespace`** is selected for *Parent Node*)	Specify a local name of the parent element of the Signature element. This attribute is only relevant for Enveloped XML Signature case. The Signature element is added at the end of the children of the parent.
Parent Node Namespace (only if the option **`Enveloped XML Signature`** is selected for the attribute *Signature Type* and **`Specified by Name and Namespace`** is selected for *Parent Node*)	Specify a namespace of the parent element of the Signature element. This attribute is only relevant for Enveloped XML Signature case. In the Enveloped XML Signature case, a null value is also allowed to support no namespaces. An empty value is not allowed.
XPath Expression (only if the option **`Enveloped XML Signature`** is selected for the attribute *Signature Type* and **`Specified by XPath expression`** is selected for *Parent Node*)	Enter an XPath expression for the parent node to be specified. This attribute is only relevant for Enveloped XML Signature case.
Key Info Content	Specifies which signing key information will be included in the KeyInfo element of the XML signature. You can select a combination of the following attribute values: ○ *X.509 Certificate Chain* X509Certificate elements representing the certificate chain of the signer key **Note** The KeyInfo element might not contain the whole certificate chain, but only the certificate chain that is assigned to the key pair entry. ○ *X.509 Certificate* X509Certificate element containing the X.509 certificate of the signer key ○ *Issuer Distinguished Name and Serial Number* X509IssuerSerial element containing the issuer distinguished name and the serial number of the X.509 certificate of the signer key ○ *Key Value* Key Value element containing the modulus and exponent of the public key **Note** You can use any combination of these four attribute values.
Sign Key Info	With this attribute you can specify a reference to the KeyInfo element.

For more information about the various attibutes, see the following:

http://www.w3.org/TR/xmldsig-core/

4. On the *Advanced* tab page, under *Transformation*, specify the following parameters.

Option	Description
Canonicalization Method for SignedInfo	Specify the canonicalization method to be used to transform the SignedInfo element that contains the digest (from the canonicalized XML document).
Transform Method for Payload	Specify the transform method to be used to transform the inbound message body before it is signed.

For more information about the various methods, see the following:

http://www.w3.org/TR/2001/REC-xml-c14n-20010315

http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments

http://www.w3.org/2001/10/xml-exc-c14n#

http://www.w3.org/2001/10/xml-exc-c14n#WithComments

5. On the *Advanced* tab page, under *XML Document Parameters*, specify the following parameters.

Option	Description
Reference Type	Enter the value of the type attribute of the content reference. If you enter null or empty, no reference type is created. More information: http://www.w3.org/2000/09/xmldsig#Object
Namespace Prefix	Enter a prefix for the XML signature namespace. Default value is **ds**.
Signature Id	Specifies the value of the Id attribute of the Signature element. If you specify no value, no Id attribute is added to the Signature element.
Output Encoding	Select an encoding scheme for the output XML document. The encoded output document will be written into the message header. If no encoding scheme is specified, the output will be *UTF-8*-encoded.
Exclude XML Declaration	Specify whether the XML declaration header shall be omitted in the output XML message.
Disallow DOCTYPE Declaration	Specify whether DTD DOCTYPE declarations shall be disallowed in the incoming XML message.

6. Save the changes.
7. To save the configurations of the signer as a template, choose *Save as Template* from the context menu of the *Signer* element.

> **Note**
>
> When you save the configuration of the *Signer* element as a template, the tool stores the template in the workspace as <ElementTemplate>.fst.

2.2.6.6.4 Verifying the PKCS#7/CMS Signature

Context

You perform this task to ensure that the signed message received over the cloud is authentic.

Working with PKCS#7/CMS Verifier

In the integration flow model, you configure the *Verifier* by providing information about the public key alias, and whether the message header or body is Base64-encoded, depending on where the Signed Data is placed. For example, consider the following two cases:

1. If the Signed Data contains the original content, then in the Verifier you provide the Signed Data in the message body.
2. If the Signed Data does not contain the original content, then in the Verifier you provide the Signed Data in the header `SapCmsSignedData` and the original content in the message body.

The Verifier uses the public key alias to obtain the public keys of type DSA or RSA that are used to decrypt the message digest. In this way the authenticity of the participant who signed the message is verified. If the verification is not successful, the verifier informs the user by raising an exception.

Under *Public Key Alias* you can enter one or multiple public key aliases for the Verifier.

> **Note**
>
> In general, an alias is a reference to an entry in a keystore. A keystore can contain multiple public keys. You can use a public key alias to refer to and select a specific public key from a keystore.

You can use this attribute to support the following use cases:

- Management of the certificate lifecycle. Certificates have a certain validity period. Using the *Public Key Alias* attribute in the Verifier step, you can enter both an alias of an existing certificate (which will expire within a certain time period) and an alias for a new certificate (which does not necessarily have to exist already in the keystore). In this way, the Verifier is configured to verify messages signed by either the old or the new certificate. As soon as the new certificate has been installed and imported into the keystore, the Verifier refers to the new certificate. In this way, certificates can be renewed without any downtime.
- You can use different aliases to support different signing senders with the same Verifier step. Using the *Public Key Alias* attribute, you can specify a list of signing senders.

> **Note**
>
> Exceptions that occur during runtime are displayed in the *Message Processing Log* view of the *Operations for SAP HANA Cloud Integration* perspective.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. Right-click a connection within the pool and choose *Add Security Element* > *Signature Verifier*.
3. In the *Model Configuration* editor, select *Verifier*.

> **Note**
>
> By default, the *Verifier* is of type PKCS#7/CMS. If you want to work with XML Signature Verifier, then choose the *Switch to XML Signer Verifier* option from the context menu.

4. In the *Properties* view, enter the details to verify the signatures of the incoming message.
5. Save the changes.

2.2.6.6.5 Verifying the XML Digital Signature

Context

The XML signature verifier validates the XML signature contained in the incoming message body and returns the content which was signed in the outgoing message body.

> **Note**
>
> For enveloping and eveloped XML signatures the incoming message body must contain only one XML signature.

The verifier supports enveloping and enveloped XML signatures and detached XML signatures. In the enveloping XML signature case, one reference whose URI references the only allowed 'Object' element via ID, and an optional reference to the optional KeyInfo element via ID is supported.

Working of XML Signature Verifier

In the validation process, a public key is required and it is fetched from the worker node keystore. On receiving the XML message, the verifier canonicalizes the data identified by the 'Reference' element and then digests it to give a digest value. The digest value is compared against the digest value available under the 'Reference' element of the incoming message. This helps to ensure that the target elements were not tampered with.

Then, the digest value of the canonicalized 'SignedInfo' is calculated. The resulting bytes are verified with the signature on the 'SignedInfo' element, using the sender's public key. If both the signature on the 'SignedInfo' element and each of the 'Reference' digest values verify correctly, then the XML signature is valid.

> **Note**
>
> XML signature verifier supports both enveloping and enveloped signatures.

Verifying Enveloping Signature: If the incoming message has enveloping signature

- an optional 'Reference' to the optional 'KeyInfo' element via ID is supported
- 'References' can have one optional transform whose algorithm must be a canonicalization method

So it implies, an enveloping XML signature with only the following structure is supported ("?" denotes zero or one occurrence, the brackets [] denotes variables whose values can vary).

```
<Signature>
      <SignedInfo>
             <CanonicalizationMethod>
             <SignatureMethod>
             <Reference URI="#[object_id]">
                    (<Transform Algorithm=[canonicalization method]/>)?
                   <DigestMethod>
                   <DigestValue>
             </Reference>
             (<Reference URI="#[keyinfo_id]">
                   (<Transform Algorithm=[canonicalization method]/>)?
                   <DigestMethod>
                   <DigestValue>
             </Reference>)?
      </SignedInfo>
      <SignatureValue>
      (<KeyInfo (Id="[keyinfo_id]")?>)?
      <Object Id="[object_id]"/>
</Signature>
```

Verifying Enveloped Signature: If the incoming message has enveloped signature

- One reference with empty URI and an optional reference to the KeyInfo element via its ID is allowed
- An additional transform containing a canonicalization method is supported, beside the transform with algorithm "http://www.w3.org/2000/09/xmldsig#enveloped-signature".

So it implies, an enveloped XML signature with only the following structure is supported ("?" denotes zero or one occurrence, the brackets [] denotes variables whose values can vary).

```
<[parent]>
       <Signature>
           <SignedInfo>
             <CanonicalizationMethod>
             <SignatureMethod>
             <Reference URI="">
                    <Transform Algorithm="http://www.w3.org/2000/09/
xmldsig#enveloped-signature"/>
                    (<Transform Algorithm="[canonicalization method]"/>)?
                   <DigestMethod>
                   <DigestValue>
             </Reference>
             (<Reference URI="#[keyinfo_id]">
                   (<Transform Algorithm="[canonicalization method]"/>)?
                   <DigestMethod>
                   <DigestValue>
             </Reference>)?
           </SignedInfo>
           <SignatureValue>
           (<KeyInfo (Id="[keyinfo_id]")?>)?
       </Signature>
</[parent]>
```

Verifying Detached Signature If the incoming message has detached signature:

- Signature element must be a sibling of the signed element
- Signed element must have an attribute of type ID
- Signature references signed element via the ID value

So it implies, a detached XML signature with only the following structure is supported ("?" denotes zero or one occurrence, the brackets [] denotes variables whose values can vary).

```
(<[signed element] Id="[id_value]">
    <!-- signed element must have an attribute of type ID -->
    ...

    </[signed element]>
    <other sibling/>* <!-- between the signed element and the corresponding
signature element, there can be other siblings -->
    <Signature>
        <SignedInfo>
            <CanonicalizationMethod>
            <SignatureMethod>
            <ReferenceURI="#[id_value]"type="[optional_type_value]">
            <!-- reference URI contains the ID attribute value of the signed
element -->
                <TransformAlgorithm=[canonicalization method]/>
                <DigestMethod>
                <DigestValue>
            </Reference>
            (<ReferenceURI="#[generated_keyinfo_Id]">
                <TransformAlgorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315" />
                <DigestMethod>
                <DigestValue>
            </Reference>)?
    </SignedInfo>
    <SignatureValue>
    (<KeyInfoId="[generated_keyinfo_id]">)?
    <Signature>)+
```

Using the *Public Key Alias*, you can enter one or multiple public key aliases for the Verifier.

> **Note**
>
> In general, an alias is a reference to an entry in a keystore. A keystore can contain multiple public keys. You can use a public key alias to refer to and select a specific public key from a keystore.

You can use this attribute to support the following use cases:

- Management of the certificate lifecycle: Certificates have a certain validity period. Using the *Public Key Alias* attribute in the Verifier step, you can enter both an alias of an existing certificate (which will expire within a certain time period) and an alias for a new certificate (which not necessarily has to exist already in the keystore). That way, the Verifier is configured to verify messages signed by either the old or the new certificate. As soon as the new certificate has been installed and imported into the keystore, the verifier refers to the new certificate. That way, certificates can be renewed without any downtime.
- You can use different aliases to support different signing senders with the same Verifier step. Using the Public Key Alias attribute, you can specify a list signing senders.

Select the *Check for Key Info Element* option to check the XML Signature for the KeyInfo element.

The KeyInfo element has to contain either the certificate chain, the certificate, the Issuer Distinguished Name and Serial Number, or the Key Value elemet (or combinations of these attributes).

> **Note**
>
> In case multiple public key aliases are specified (using the *Public Key Alias* attribute), this option is mandatory (to make sure that from the KeyInfo the public key can be derived).

Procedure

1. In the *Model Configuration* editor, select *Verifier* element and from the context menu, select *Switch to XML Signature Verifier* option.
2. In the *XML Signature Verifier* tab page, enter the parameters to verify XML digital signature for the incoming message.
3. Save the changes.

2.2.6.6.6 Encrypting and Signing the Message Content with PKCS#7/CMS

Context

You perform this task to protect the message content from being altered while it is being sent to other participants on the cloud, by encrypting the content. In the integration flow model, you configure the *Encryptor* by providing information on the public key alias, content encryption algorithm, and secret key length. The encryptor uses one or more receiver public key aliases to find the public key in the keystore. The encryption process uses a symmetric key of specified length for content encryption. The symmetric key is encrypted by the public recipient key with the cipher. The encryption is determined by the type of *Content Encryption Algorithm* that you select. The encrypted content and the receiver information containing the symmetric encryption key are placed in the message body.

In addition to encrypting the message content, you can also sign the content to make your identity known to the participants and thus ensure the authenticity of the messages you are sending. This task guarantees your identity by signing the messages with one or more private keys using a signature algorithm.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. Right-click a connection within the pool and choose *Add Security Element* > *Content Encryptor*.
3. In the *Model Configuration* editor, select *Encryptor*.
4. To configure the encryptor with a saved configuration that is available as a template, choose *Load Element Template* from the context menu of the *Encryptor* element.
5. In the *Properties* view, specify the general settings for encryption.

Option	Description
Block Size (in Bytes)	Enter the size of the data that is to be encoded. If you enter a value equal to or less that **0**, the whole data is encoded.
Encode Body with Base64	Select this option if the message body will be base64-encoded.

Option	Description
Signatures in PKCS7 Message	Select one of the following options: ○ Enveloped Data Only Select this option if you want to apply encryption only. ○ Signed and Enveloped Data Select this option if you want to apply both encryption and signing.

6. Specify the settings for the encryption process (under *Encryption*).

Option	Description
Content Encryption Algorithm	Specify the algorithm that is to be used to encrypt the payload.
Secret Key Length	Specify the key length. The offered key lengths depend on the chosen encryption algorithm.
Receiver Public Key Alias	Specify one or more aliases. You use the alias to select the public key from the keystore.

7. Specify the settings for the signing process under *Signature* (only if you selected `Signed and Enveloped Data` for *Signatures in PKCS7 Message*).

   For each private key alias (specified under *Signer Parameters*), you define the following parameters:

Option	Description
Signature Algorithm	Specify the signature (digest) algorithm.
Include Certificates	If you activate this option (value true), the signer's certificate chain will be added to the SignedAndEnvelopedData object of the message.

8. Save the changes.
9. To save the configuration of the encryptor as a template, choose *Save as Template* from the context menu of the *Encryptor* element.

> **Note**
>
> When you save the configuration of the *Encryptor* element as a template, the tool stores the template in the workspace as <ElementTemplateName>.fst.

2.2.6.6.7 Encrypting the Message Content with OpenPGP

You have the option to encrypt the message content using Open Pretty Good Privacy (OpenPGP).

Context

You have the following options to protect communication on message level based on the OpenPGP standard:

- You can only encrypt a payload.

- You can sign and encrypt a payload.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. Right-click on a connection within the pool and choose *Add Security Element* > *Content Encryptor*.
3. In the *Model Configuration* editor, position the cursor on the *Content Encryptor* step and in the context menu select *Switch to PGP*.
4. Open the *Properties* view.
5. Specify whether or not to sign the payload (in addition to applying payload encryption). To do that, select one of the following options for *Signatures in PGP Message*.

Option	Description
Include Signature	Select this option if you want to apply both encryption and signing.
No Signature	Select this option if you want to apply encryption only.

6. Under *Encryption*, specify the following parameters for message encryption:

Option	Description
Content Encryption Algorithm	Specify the algorith which is to be used to encrypt the payload. This is the symmetric key algorithm.
Secret Key Length	Specify the key length. The offered key lengths depend on the chosen encryption algorithm.
Compression Algorithm	Specify the compression algorithm. There is the option to apply compression before the encryption step. In case you do not want to apply compression, select *Uncompressed*.
Armored	Select this option in case the output shall be radix 64- (base64-) encoded with an additional header. Selecting this option increases the message security level and, therefore, is recommended.
With Integrity Packet	Select this option if you want to create an Encrypted Integrity Protected Data Packet. This is a specific format where an additional hash value is calculated (using the SHA-1 algorithm) and added to the message. Selecting this option increases the message security level and, therefore, is recommended.
Encryption Key User IDs of the Public Keyring	You can specify the encryption key user IDs (or parts of it). Based on the encryption key user ID, the public key (for message encryption) is looked up in the PGP public keyring. You can specify multiple user IDs. Furthermore, you can select multiple user IDs by specifying

7. Under *Signature*, specify the following parameters for message signing (only in case you have selected `Include Signature` for *Signatures in PGP Message*).

Option	Description
Digest Algorithm	Specify the digest algorithm (or: hash algorithm) that is to be applied.

Option	Description
	The sender calculates out of the message content a digest (or hash value) using a digest algorithm. To finalize the payload signing step, this digest is encrypted by the sender using a private key.
Signer Key User ID Parts of Keys from the Secret Keyring	Specify the signer key user ID parts. Based on the signer key user ID parts, the private key (for message signing) is looked up in the PGP secret keyring.

8. Save the changes.

2.2.6.6.8 Decrypting and Verifying the Message Content with PKCS#7/CMS

Context

You perform this task to decrypt messages received from a participant on the cloud.

Note that the related keystore must contain the private key, otherwise decryption of the message content will not work.

It is also possible to verify the signature of a SignedAndEnvelopedData object to ensure that the received signed message is authentic.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. Right-click a connection within the pool and choose *Add Security Element* > *Content Decryptor*.
3. In the *Model Configuration* editor, select *Decryptor*.
4. In the *Properties* view, enter the details to decrypt the content of the incoming message.

Option	Description
Name	Name of the selected PKCS7 Decryptor element. The deafult name is PKCS7Decryptor. If you add a second PKCS7 Decryptor to the integration flow, you have to change the name because the Decryptor name must be unique.
Body is Base64-Encoded	Select this option if a Base64-encoded message body is expected.
Signatures in PKCS7 Message	Specify which kind of content can be processed.

Option	Description
	You can select one of the following options: ○ Enveloped Data Only Encrypted-only payloads can be processed. ○ Signed and Enveloped Data Payloads with content type *Signed and Enveloped Data* can be processed. ○ Enveloped or Signed and Enveloped Data Both content types – Enveloped Data Only or Signed and Enveloped Data - can be processed.

5. Depending on the option you have chosen for *Signatures in PKCS7 Message*, you can specify further settings (only if you have specified **`Enveloped or Signed and Enveloped Data`** or **`Signed and Enveloped Data`** for *Signatures in PKCS7 Message*):

Option	Description
Public Key Alias	Enter the public key aliases of all expected senders. These are the public key aliases corresponding to the private keys (of the expected senders) that are used to **sign** the payload. The public key aliases specified in this step restrict the list of expected senders and, in this way, act as an authorization check.

The public key alias list specified in this step is only required for signed content. The private key required to decrypt the payload automatically is found by the system; it is the private key deployed on the corresponding tenant.

Signed content is verified as follows: SAP HCI checks the list of aliases as specified in this step. As soon as a signature is found that is signed by a private key corresponding to one of the public keys listed in the alias list, the signature is verified by the public key. If the verification is successful, the payload is accepted. No further check is executed.

6. Save the changes.

2.2.6.6.9 Decrypting the Message Content with OpenPGP

You have the option to decrypt the message content using Open Pretty Good Privacy (OpenPGP).

Context

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. Right-click on a connection within the pool and choose *Add Security Element* > *Content Decryptor*.
3. In the *Model Configuration* editor, position the cursor on the *Content Decryptor* step and in the context menu select *Switch to PGP*.
4. Open the *Properties* view.
5. Specify the following parameters for message decryption:.

Option	Description
Name	Name of the selected PGP Decryptor element. The deafult name is PGPDecryptor. If you add a second PGP Decryptor to the integration flow, you have to change the name because the Decryptor name must be unique.
Signatures in PGP Message	Specify the expected payload content type which is to be decrypted. You have the following options: ○ No Signatures Expected When you select this option, the decryptor expects an inbound message that doesn't contain a signature. **Note** If you select this option, inbound messages that contain a signature cannot be processed. ○ Signatures Required When you select this option, the decryptor expects an inbound message that contains a signature. ○ Signatures Optional When you select this option, the decryptor can process messages that eitehr contain a signature or not.
Signer Key User ID Parts of Keys from the Public Keyring	Specify the signer key user ID parts of all expected senders. Based on the signer key user ID parts, the public key (for message verification) is looked up in the PGP public keyring. The signer key user ID parts specified in this step restrict the list of expected senders and, in this way, act as an authorization check.

6. Save the changes.

2.2.6.7 Defining Service Call

Prerequisites

You have added the service call element to the integration flow model from the palette.

Context

Service Call is used to call an external system. Such calls enable transaction of data from or to the target system. It can be used for following types of operations:

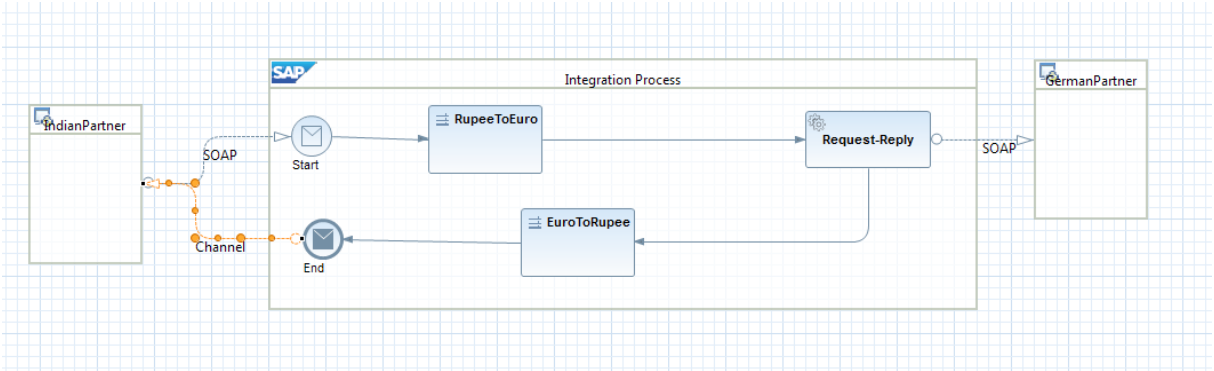
1. Request-Reply
2. Content Enrichment

2.2.6.7.1 Defining Request-Reply

Context

You can use this task to enable request and reply interactions between sender and receiver systems.

Example:



Suppose currency conversion is required for a transaction between Indian and German business partners. In such a scenario, using Request-Reply pattern, user can successfully convert Indian currency to German currency and vice versa. As shown in the integration flow, the request mapping converts the currency in payload to Euro and reply mapping converts the currency in payload back to Indian Rupee.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. If you want to add a *Request-Reply* element in the integration flow, choose *Add Service Call* from the context menu of a connection within the pool.
3. From the *Palette*, select *Others* > *Receiver*, and drop it outside the *Integration Process* pool.
4. From the speed button of the *Request-Reply* element, create a connection that represents a channel from the *Request-Reply* element to the *Receiver*.
5. Configure the channel with the required adapter details.
6. You can add integration flow elements between the *Request-Reply* and *End Message* element, for processing response payload from *Receiver* system.

7. You can also connect *End Message* element back to the sender. Also, ensure that no adapter is configured for this channel (as highlighted in the integration flow screenshot).

> **Note**
>
> In this release, the channel connecting the external system with the Request-Reply element can be configured with the SOAP, SuccessFactors, HTTP and IDOC adapters.

2.2.6.7.2 Defining Content Enricher

Context

You use this task to enrich an incoming message by appending the additional data retrieved from an external resource before it is sent to the target receiver. Content Enricher offers inbuilt aggregation mechanism to merge incoming payload with content fetched from external resource.

> **Example**
>
> Suppose a message contains job description details, such as Job ID, Job Category and Job Location. Whereas, the target receiver requires additional information such as the profile details of the candidate who registers for that job and the hiring manager details.
>
> In such a scenario, the incoming message can be enriched twice to get the desired output message. The first content enricher will append the profile details of the candidate after the job description details. This enriched output message of the first content enricher acts as the input message to the second content enricher, which appends the hiring manager details to the previously enriched message. In this way, the original message is enriched with the required information as expected by the receiver.

The content enricher uses a look up channel to retrieve additional data from an external resource and it appends the data to the incoming message as shown below:

```
<n1:Messages xmlns:multimap="http://sap.com/xi/XI/SplitAndMerge">
   <n1:Message1>
      [XML document 1]
   </n1:Message1>
   <n1:Message2>
      [XML document 2]
   </n1:Message2>
   <n1:Message3>
      [XML document 3]
   </n1:Message3>
    ...
</n1:Messages>
```

> **Note**
>
> Data Type Definitions (DTDs) that might exist in the original XML or in the second looked-up XML will not be part of the enriched XML.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. Choose *Add Service Call* from the context menu of a connection within the pool.
3. Right click on *Request -Reply* element and select *Switch To Content Enricher*.
4. From the *Palette*, select *Others* > *Reciever*, and drop it outside the *Integration Process* pool.
5. From the speed button of the *Receiver* element, create a connection that represents a channel from the *Receiver* to the *Content Enricher*.
6. Configure the channel with the SuccessFactors connector details.

> **Note**
>
> In this release, the channel connecting the external resource with the content enricher can be configured with the SuccessFactors adapter and SOAP 1.x.

2.2.6.8 Defining Error Handling Strategy

Context

You use this task to handle errors when message processing fails in the runtime. You select an error handling strategy based on the description below:

- *Raise Exception*- When a message exchange could not be processed, an exception is raised back to the sender.
- *None*- When a message exchange could not be processed, no error handling strategy is used.

Procedure

1. In the *Model Configuration* editor, click on the graphical area outside the integration flow model.
2. In the *Properties*view, select the *Error Configuration* tab.
3. From the *Error Handler* strategy dropdown, select one of the options- *Raise Exception* or *None*.

2.2.6.9 Specifying Runtime Configuration

Context

Specify the runtime configurations for the integration flow that are required when the integration flow is executed in runtime. You configure the following parameters:

- **Namespace Mapping**: Maps a prefix to a namespace at runtime. This parameter is required for elements such as content-based router, content modifier or content filter that can use namespaces in their configuration. You can either enter the namespace-prefix pair or the tool automatically fills the namespace-prefix pair whenever a lookup is performed for the assigned WSDL.

  > Example
  >
  > See the namespace mapping example below:
  >
  > xmlns:ns0=https://hcischemas.netweaver.neo.com/hciflow
  >
  > Here, **ns0** is the prefix and **https://hcischemas.netweaver.neo.com/hciflow** is the namespace. So, in a gateway, you can specify the routing condition for an XML message as:
  >
  > ```
  > /ns0:HCIMessage/SenderID='Sender01'
  > ```

  To know where you need to specify namespace for content-based router, content modifier or content filter flowsteps, see the relevant step in the corresponding tasks Defining Gateway, Defining Content Modifier, Defining Content Filter.

- **Message Processing Log**: Allows you to configure the log level to display the logs in the *Log* view of the *Message Monitoring* editor in the *Integration Operations* perspective.
- **Allowed Header(s)**: Allows you to specify the headers to be retained when the incoming message is processed. In general, headers used by the runtime components are retained and other headers are removed during message processing. This field is useful when you need specific header information along with the message body.

Procedure

1. In the *Model Configuration* editor page, select the graphical area outside the integration flow model.
2. In the *Properties* view, choose the *Runtime Configuration* tab.
3. In the *Namespace Mapping* field, enter a namespace-prefix pair with a format **`xmlns:<prefix>=<namespace>`** or select it from the dropdown.

   > Note
   >
   > The field is required if the integration flow contains content-based routing. You can also enter multiple namespace-prefix as shown in the example- **`xmlns:test=http://sapcd.com/testABC;xmlns:test2=http://sapcd.com/testPQR`**

4. Select one of the log levels from the *Message Processing Log* dropdown.

   > **Note**
   >
   > The available logging levels are:
   >
   > - *All Events*: Allows all messages to be logged and displayed
   > - *Error Events*: Allows only error messages to be logged and displayed
   > - *No Logging*: Hides logging information

5. In *Allowed Header(s)* field, enter one or more names of the headers by separating them with pipe (|).
6. Select *Enable Debug Trace* option to enable server log in case you need to obtain help from SAP Support for troubleshooting issues.

   > **Caution**
   >
   > If you select *Enable Debug Trace* option and if the SaaS Admin has set the log level for `org.apache.cxf` as `INFO`, the message payload is written into the log files. If the message payload contains sensitive data, then you should consider the settings as the message payload should not be logged.

7. Save the changes.

Related Information

Defining Gateway [page 86]
Defining Content Filter [page 73]
Defining Content Modifier [page 68]

2.2.6.10 Configuring Message Storage at a Process Step

Context

You use this task to configure a process step to store a message payload so that you can access the stored message and analyze it at a later point in time.

In the integration flow, you mark a process step for persistence by specifying a unique step ID, which can be a descriptive name or a step number. For example, a step ID configured after a mapping step could be *MessageStoredAfterMapping*.

At runtime, the runtime node of the cluster stores information, such as GUID, MPL GUID, tenant ID, timestamp, or payload, for the messages at the process steps that have been marked for persistence.

The message storage feature is useful for the following cases

- **Auditing**: The SaaS Admin can use this feature to analyze messages that have been processed.

  > **Note**
  >
  > Messages can be stored in the runtime node for 90 days, after which the messages are automatically deleted.

- **Providing access to data owned by the customer** : Since SAP HANA Cloud Integration processes messages on behalf of the customer, you might need to access the customer-owned data after the messages have been processed to provide it to the customer.

Procedure

1. Open the <integration flow>.iflw on the *Model Configuration* editor page.
2. Determine if you want to store the message payload before or after a process step.

   For example, you can store the message payload before and after a mapping step, so you add Persist Message at the two points.
3. In the *Model Configuration* editor, right-click the connection at the determined point within the pool and choose the *Add Persist Message* option.
4. In the *Model Configuration* editor, select the Persist Message element.
5. In the *Properties* view, retain the auto-generated step ID for the process step, or edit it if required.
6. If you want to encrypt the message payload that is stored at the step ID, choose *Encrypt the stored message payload*.
7. Save the changes.

2.2.6.11 Scheduling a Process Start

Context

If you want to configure a process to automatically start in a particular schedule, you can use this procedure to set the date and time on which the process should run once or repetitively. The day and time combinations allow you to configure the schedule the process requires. For example, you can set the trigger just once on a specific date or repetitively on that date, or you can periodically trigger the timer every day, specific days in a week or specific days in a month along with the options of specific or repetitive time.

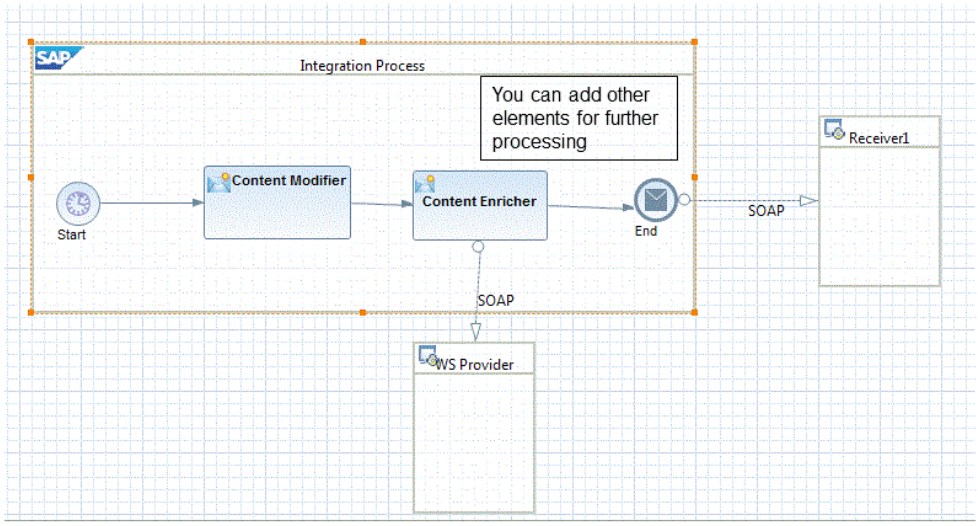
Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.

2. From the *Palette*, expand the *Event* category.
3. Select the *Timer Start* notation and drop it within the *Integration Process* pool such that it is the first element of the integration flow.
4. From the speed button of the *Timer Start*, select the sequence flow and connect to the following element within the *Integration Process* pool of the integration flow.
5. Select the *Timer Start* and in the *Properties* view, enter the details for setting the schedule or specific day in which the process should start.
6. Save the changes

Example

The image below shows a sample scenario that requires to periodically poll a WebService endpoint to fetch messages and send them to a receiver.



2.2.7 Defining Exception Subprocesses

Context

You use this task if you want to catch any exceptions thrown in the integration process and process them.

Procedure

1. Open the <integration flow>.iflw in the *Model Configuration* editor.
2. To add an exception subprocess to the integration flow, choose *Others* > *Exception Subprocess* from the palette. The subprocess can be dropped into the integration process and should not be connected to any of the elements of the integration flow.
3. Select the exception subprocess you have added to the integration flow model, to configure it.
4. The process should always start with an *Error Start* event.
5. There can only be one end event. It can be either an *End Message* or an *Error End* event.

> **Note**
>
> - You can use an *End Message* event to wrap the exception in a fault message and send it back to the sender in the payload.
> - You can use an *Error End* event to throw the exception to default exception handlers.

6. You can add various tasks between the start and end events. For example, you can choose *Add Service Call* from the context menu of a connection within the pool. This enables you to call another system to handle the exception.
7. Save the changes.

> **Note**
>
> - The message processing log will be in an error state even if a user catches an exception and performs additional processing on it.
> - You cannot enable exception subprocess for integration flow scenarios containing the timer element.
> - You can get more details on exception using `${exception.message}` or `${exception.stacktrace}`.
> - You cannot configure the following integration flow elements within an exception subprocess:
>   - Splitter
>   - Router
>   - Multicast
>   - Join
>   - Gather

2.2.8 Checking the Consistency

Context

You can execute consistency checks to validate if the configurations of the integration flow elements are adhering to their definition. This helps you validate the integration flow model and the configured attributes are

supported at runtime. The inconsistencies can occur if the integration flow model does not adhere to the modeling constraints that SAP supports for specific scenarios. When you trigger consistency check execution, the tool validates the consistency of your integration flow model against the predefined constraints set by the SAP supported scenarios.

Procedure

1. Right-click on the project and choose *Execute Checks*.

   > **Tip**
   >
   > You can also click on the graphical model outside the Integration Process pool, and choose *Execute Checks* from the context menu.

2. Open the *Problems* view for the consistency check results.

2.2.9 Saving Integration Flow as a Template

Prerequisites

You have specified the location path to store the user-defined templates at *Windows* > *Preferences* > *SAP HANA Cloud Integration* > *Personalize*.

Context

You perform this task to save an integration flow as a template. The integration flow saved as template retain the attribute values so that you can reuse the template for similar configuration scenarios.

Procedure

1. In the *Project Explorer* view, open the <integration flow>.iflw from your project.
2. From the main menu, choose *File* > *Save As...*.
3. In the *Save As Integration Flow Template* dialog, enter a name for the template.
4. If you want to retain the externalized parameters in the template then select *Retain the externalized parameter(s)* option.

> **Note**
>
> If the integration flow contains externalized parameters and you do not select this option, the integration flow gets saved with the most recent values you have assigned to the externalized parameters.

5. Choose *OK*.

Results

You can find the new template in the location you have mentioned in the *Preferences* page. When you create the integration flow using the template, you can find the saved configurations.

2.3 Developing Value Mappings

2.3.1 Creating a Value Mapping Project

Context

You use this task to create a value mapping definition that represent multiple values of an object. For example, a product in Company A is referred by the first three letters as 'IDE' whereas in Company B it is referred by product code as ''0100IDE''. When Company A sends message to Company B, it needs to take care of the difference in the representations of the same product. So Company A defines an integration flow with a *Mapping* element and this *Mapping* element contains reference to the value mapping definition. You create such value mapping groups in a *Value Mapping* project type. You can also create value mapping definitions from the web UI application. For more information, see Working with Value Mappings [page 163]

> **Note**
>
> You can deploy only one value mapping project on the tenant.

Procedure

1. In the main menu, choose *File* > *New* > *Project...*.
2. In the *New Project* wizard, select *SAP HANA Cloud Integration* > *Integration Project*.
3. Choose *Next*.

4. Enter a project name and select the required *Project Type* as *Value Mapping*.
5. Choose *Finish*.

   A new project is available in the *Project Explorer* view. By default, the new value mapping project is configured with IFLMAP runtime node type.

2.3.2 Editing the Value Mapping Project

Context

You use this task to define value mapping groups in the *value_mapping.xml* file that is available under the *Value Mapping* project type. You enter the group IDs for each set of agency, schema and a value of the object that together identify a specific representation of the object.

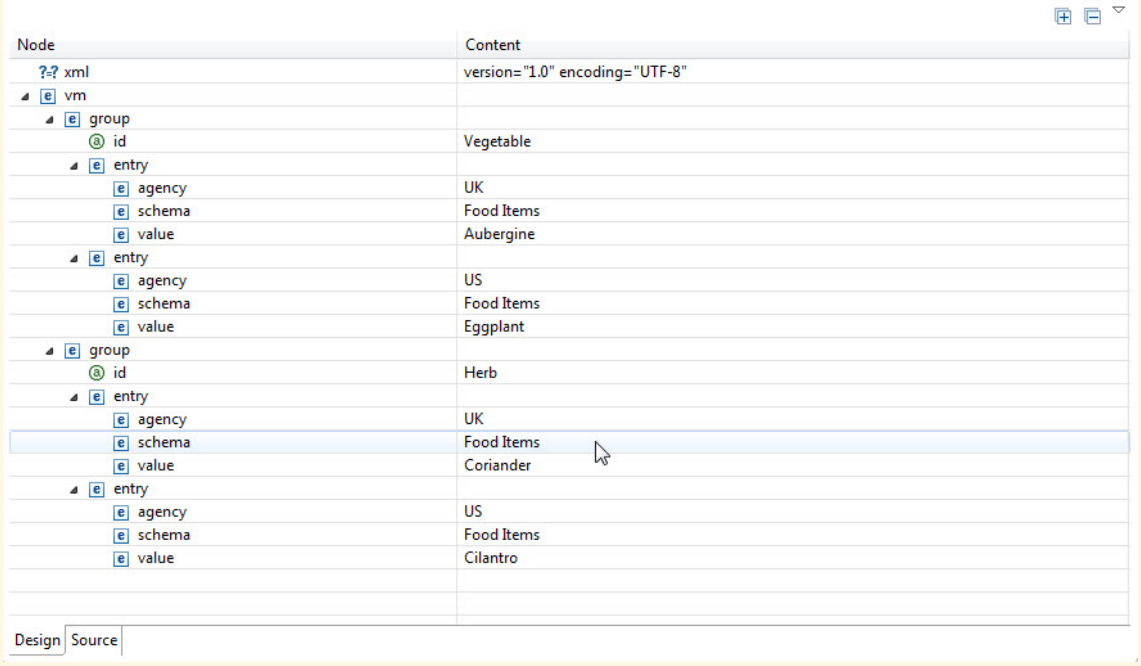
Procedure

1. Open the *value_mapping.xml* in the editor and choose the *Source* tab page editor.
2. Enter the group ID, agency, schema and value as shown in the example below, and save the changes.

```
<?xml version="1.0" encoding="UTF-8"?>
<vm>
    <group id="Vegetable">
        <entry>
            <agency>UK</agency>
            <schema>Food Items</schema>
            <value>Aubergine</value>
        </entry>
        <entry>
            <agency>US</agency>
            <schema>Food Items</schema>
            <value>Eggplant</value>
        </entry>
    </group>
    <group id="Herb">
        <entry>
            <agency>UK</agency>
            <schema>Food Items</schema>
            <value>Coriander</value>
        </entry>
        <entry>
            <agency>US</agency>
            <schema>Food Items</schema>
            <value>Cilantro</value>
        </entry>
    </group>
</vm>
```

Tip

If you want to edit the values, you can switch to the *Design* tab page editor.



2.3.3 Exporting and Importing Value Mapping Groups

Context

You use this task to either import a .csv file containing value mapping groups into the *Value Mapping* project type within your workspace or export the content of *value_mapping.xml* from your workspace and store it as a .csv file in your file system. The format of the valid .csv file containing value mapping groups is shown in the image below:

	Agency1|Schema1	Agency2|Schema2
GroupID1	Value_A	Value_a
GroupID2	Value_B	Value_b

This task shows the steps for a simple scenario that requires you to export value mappings from your workspace, and import the same value mappings into a workspace located in another system.

Procedure

1. Export the value mapping groups into your local file system
   a. In the *Project Explorer*, select the value mapping project and choose *Export Value Mappings*.
   b. In the *Export Value Mapping Groups* wizard, select the required value mapping groups and choose *Next*.
   c. In the *Export groups into a file* page, enter a name for the .csv file and browse for a location to store the file.
   d. Choose *Finish*.

   The .csv file containing the exported value mapping groups is available at the selected file location.

   Example

   The image below shows an example of a vale mapping group exported into a .csv file.

	A	B	C
1		US|Food Items	UK|Food Items
2	Herb	Cilantro	Coriander
3	Vegetable	Eggplant	Aubergine
4			

2. Import the value mapping groups
   a. In the *Project Explorer*, select the value mapping project that you want to import the value mappings to.
   b. Choose *Import Value Mappings*.
   c. In the *Select a CSV File* page, browse for the .csv file.
   d. Choose *Finish*.

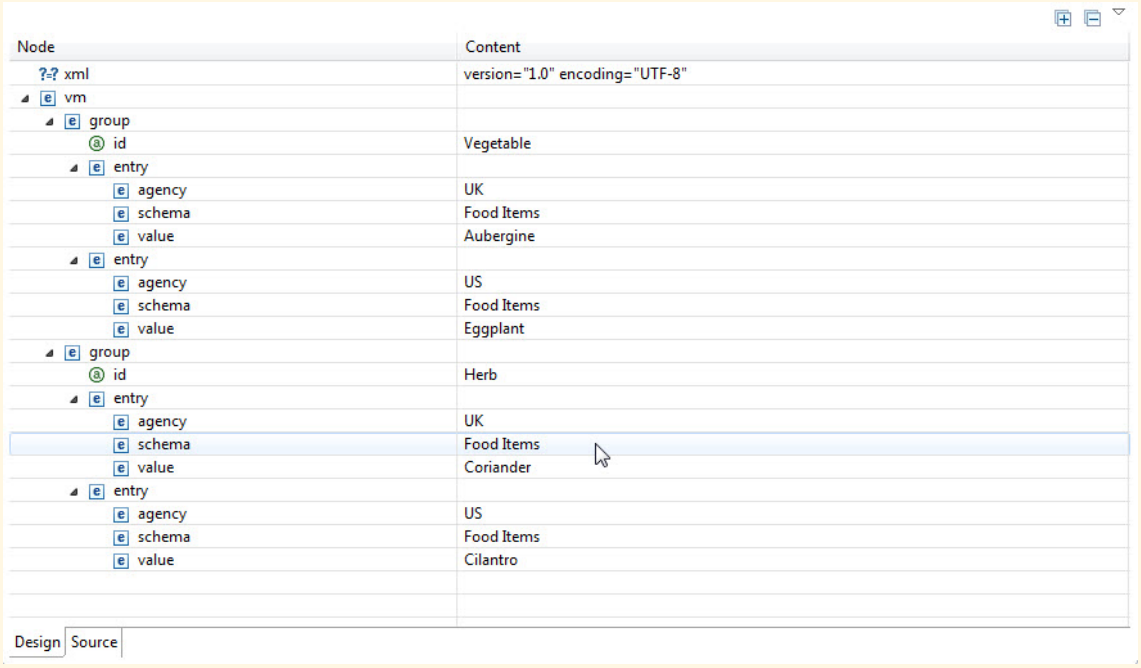
   Note

   You cannot import value mappings that have been exported from Eclipse. If you do so, then the existing version of the value mapping files changes.

   The .csv file is imported as *value_mapping.xml* file, and is available under the value mapping project.

Example

The screenshot below shows how the content of the .csv file (as shown in the previous screenshot) gets imported as the *value_mapping.xml*.



2.3.4 Referencing Value Mappings from a Message Mapping

Prerequisites

- You have imported message mapping (.mmap) from an On-Premise repository into the **src.main.resources.mapping** folder of the integration project in your workspace.
- You have placed the required source and target WSDLs into the **src.main.resources.wsdl** folder of the integration project in your workspace.
- You have added *value_mapping.xml* under the value mapping project.

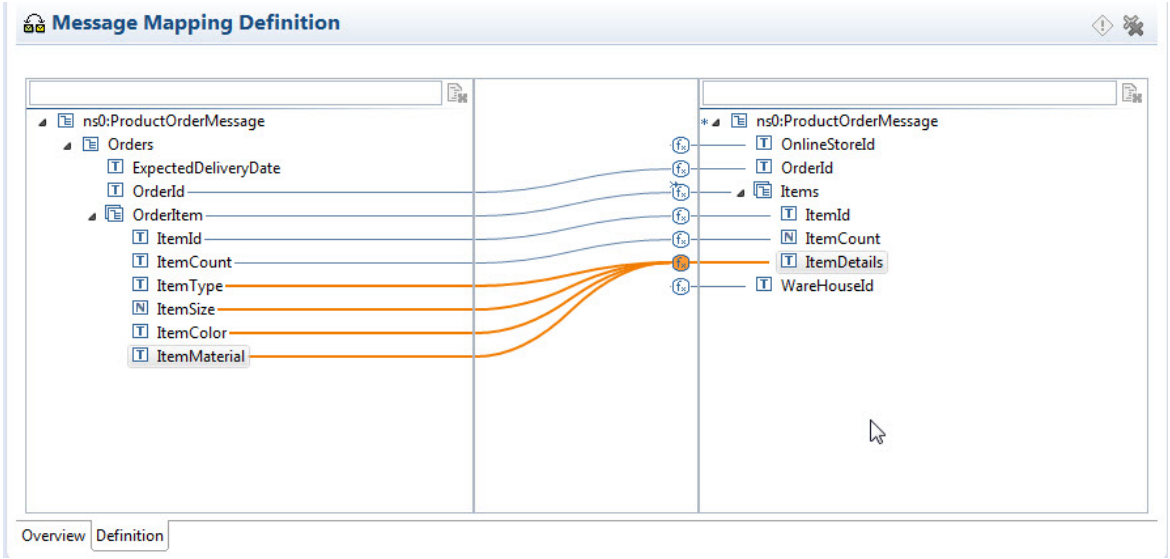
Context

You use this procedure to configure the message mapping definition with references to a value mapping. The value mapping referencing is required when a node value of a message needs to be converted to another

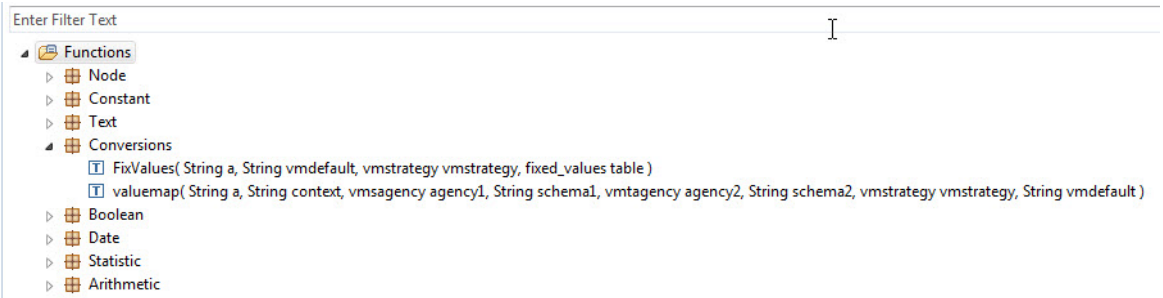
representation. The runtime can evaluate the reference only if you deploy the integration flow project containing the message mapping, and the associated value mapping project on the same tenant.

Procedure

1. In the *Project Explorer* view, expand the integration flow project.
2. Under *src.main.resources.mapping* folder, double-click the <message map>.mmap to open it in the message mapping editor.
3. In the *Message Mapping Overview*, under the *Signature* category, provide the source and target WSDLs.
4. Choose the *Definition* editor tab page to view the message mapping tree structure.
5. Connect a source node item with one or more target node item to define the mapping.
6. Double-click the function icon on the connection, denoted as f_x, to open the mapping relation between the source and target elements.
   For example, see the screenshot below:

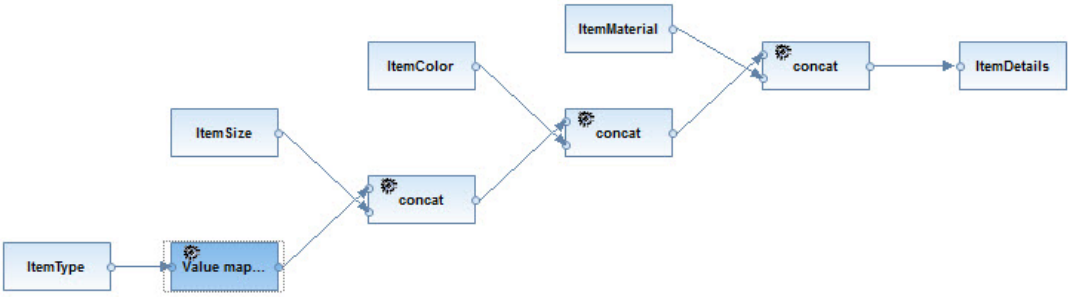


7. In the *Expression* tab page of the *Properties* view, expand the *Function* folder.
   For example, see the screenshot below:



8. Select the *valuemap* option under the *Conversion* package and drop it within the *Expression* tab page.
9. Connect the required node and the valuemapping function.

For example, see the screenshot below:



10. Double-click the value mapping function and provide details for the value mapping parameters.
11. Save the changes.

2.3.5 Checking the Value Mapping Consistency

Context

You can execute a consistency check to validate if the content of the project is adhering to the required definition of a value mapping.

The consistency check is executed on the *value_mapping.xml* file. The inconsistencies can be mainly due to invalid content entered in the *value_mapping.xml* such as the value for an agency-schema pair is repeated , incorrect tags or missing tags.

Procedure

1. Right-click on the project and choose *Execute Checks*.
2. Open the *Properties* view, and view the result of the check.

2.4 Externalizing Parameters of Integration Flow

Context

Externalizing parameters is useful when the integration content has to be used across multiple landscapes, where the endpoints of the integration flow can vary in each landscape. This method helps you to declare a parameter as a variable entity to allow you to customize the parameters on the sender and receiver side with a change in landscape.

Partial Parameterization enables to change part of a field rather than the entire field. This variable entity of the field is entered within curly braces.

List of parameters that can be externalized

- *SOAP (SAP RM) Connector*: Address, URL to WSDL
- *SOAP (SOAP 1.x) Connector*: Address, URL to WSDL
- *IDoc (IDoc-SOAP) Connector*: Address, URL to WSDL
- *SFTP Connector*: Server Host , User Name , Directory
- *Sender Authorization*: Subject DN and Issuer DN

Procedure

1. Externalize parameters by extraction.

   > **Note**
   >
   > You can extract certain parameters that are already configured in the integration flow, so that the parameters are externalized. You can view the extracted parameters in the *Externalized Parameter* view. If you have already extracted a parameter, the parameter cannot be extracted again.

   a. From the *Project Explorer* view, open the <integration flow>.iflw in the editor.
   b. In the *Model Configuration* editor, right-click the graphical area outside the model and choose *Extract Parameters* from the context menu.
   c. In the *Extract Parameters* dialog, select the parameters that have to be extracted and be available for externalization.

      > **Note**
      >
      > In the *Console* view, you can see a summary of the parameter extraction.

   d. Open the *Externalized Parameters* view.

      > **Note**
      >
      > You can view the externalized parameters of the field in the integration flow. The field is now available as a variable.

e. From the toolbar of the view, choose  (*Sync with Editor* icon) to synchronize the integration flow that is currently open with the *Externalized Parameters* view.

f. Choose  (*Reload* icon) to update the view with the list of externalized parameters.

g. You can edit parameter values and then choose  (*Save Parameters* icon), from the toolbar of the view.

> **Note**
>
> The *Save Parameters* icon is enabled only if the parameter values and the type are consistent.For drop down or combo box the values should be edited from here else, the parameter references are lost.

2. Externalize parameters by manual configuration.

> **Note**
>
> You can use the step below if you want to externalize certain parameters in such a way that you do provide any values in the integration flow but want to manually configure the attributes in the *Externalized Parameters* view.

a. In the *Project Explorer* view, open the <integration flow>.iflw in the editor.
b. In the *Model Configuration* editor, double-click the channel.
c. In the *Channel* editor, choose the *Adapter-specific* tab.
d. Enter a value for the field that you want to externalize, in this format: {{<parameter name>}}.

> **Note**
>
> You can externalize the value of the field so that the field is now a variable. For example, you might need to change the 'Address' field of a connector at the receiver without making any other changes to the integration flow. So, you enter **`{{server}}`** in the 'Address' field. The steps below show how you can specify data for the externalized parameters.
>
> Only part of the field can also be changed. For example, in the given URL, http://**`{{host}}`**:**`{{port}}`**/FlightBookingScenario, host and port are the variable entities.

e. Save the changes in the channel editor.
f. Open the *Externalized Parameters* view.
g. From the toolbar of the view, choose *Sync with Editor* (icon) to synchronize the integration flow that is currently open with the *Externalized Parameters* view.

> **Note**
>
> In the *Externalized Parameters* view, you can see all the parameters that have been externalized in the integration flow. For example, if you have externalized the 'Address' field of a connector as **`{{server}}`** or you have selected this field for extraction, you can see the parameter under the *Name* column.

h. Open the *Externalized Parameters* view and choose  (*Reload* icon).

> **Note**
>
> In the *Externalized Parameters* view, you can view all the parameters that have been externalized in the integration flow. For example, if you have externalized the 'Address' field of a connector as `{{server}}` or you have selected this field for extraction, you can see the parameter under the *Name*.

i. From the toolbar of the *Externalized Parameters* view, choose  (*Save Parameters* icon).

> **Note**
>
> - The *Save Parameters* icon in the *Externalized Parameters* view is enabled only if the parameter values and the type are consistent.
> - If you do not save the view using the *Save Parameters* icon, the *parameters.prop* file is not updated with the new externalized parameters, which is indicated with an error marker in the *Integration Flow* editor. If the editor has no unsaved changes, the error marker remains even though the parameter content is saved. In such a case, you can execute the consistency check by using the *Execute Checks* context menu option in the *Model Configuration* editor.

2.4.1 Configuring Multiple Externalized Parameters

Context

You use this task when you want to provide the same values for the common parameterized attributes across multiple integration projects. The tool offers quick parameterization with the *Configurations* view that displays a list of externalized parameters from all the projects. If multiple integrations require same values for the externalized parameters, you can choose the *Mass Parameterize* option to fill values for all the common parameters.

Procedure

1. In the main menu, choose *Window* > *Show View* > *Others*.
2. In the *Show View* dialog, choose *Others* > *Configurations*.
3. In the *Configurations* view toolbar, choose  (*Mass Configure*).
4. In the *Mass Configuration* wizard, select the projects that contain the common parameters and choose *Next*.
5. Enter values for the list of parameters and choose *Next*. These values get applied to their corresponding parameters that are common across the selected projects.

> **Note**
>
> A common parameter that has been assigned different values, is indicated using the tag *Parameter has Different Values*. If you delete this tag and enter a new value or keep it empty, the value of the common parameter changes accordingly across all the projects that use it.

6. Confirm if you want to update the common parameters with the new values.
7. If you want to view or change externalized parameters in the integration flow from the *Configurations* view, double-click any file under the *Configurations* view to open the *Model Configuration* editor and the *Externalized* view.
8. If you want to deploy the projects, select *Deploy Integration Content* and enter the *Tenant ID*.
9. Choose *Finish*.

Results

The completion of this task triggers the build automatically and your projects get deployed with the updated externalized parameters.

2.5 Deploying an Integration Project

Prerequisites

You have obtained the details about the system management node from the SaaS Admin and entered in *Windows* > *Preferences* > *SAP HANA Cloud Integration* > *Operations Server* page.

You have ensured that the supported version of Eclipse IDE, Eclipse Juno (Classic 4.2.2), is installed on your system to avoid deployment failure.

Context

You perform this task to trigger deployment of the integration project. The deployment action triggers the project build and deployment on the Tenant Management Node (Secure).

Procedure

1. If you are deploying only one integration flow project, follow the steps below:
   a. Right-click on the project and choose *Deploy Integration Content* from the context menu.

      > **Note**
      >
      > You can distinguish each deployment of the project bundle by adding a qualifier in the MANIFEST.MF file. Open the MANIFEST.MF file of the project and enter the value for **Bundle-Version** as `<version>.qualifier`. For example, `1.0.0.qualifier`.

   b. In the *Integration Runtime* dialog, enter the *Tenant ID* of the Tenant Management Node.
   c. Choose *OK*.
2. Check the deployment status of the integration project in the runtime by following the steps below:
   a. In the *Node Explorer* view, select the Tenant that is represented as  **<Tenant ID>**.
   b. Open the *Task* view and check if the 'Generate and Build' task is available.
   c. In the *Node Explorer*, select the worker node type represented as  **IFLMAP** and check for the project bundle that is deployed in the *Component Status* view.

      > **Note**
      >
      > You can identify the project bundle by checking the *Version* column. If you have specified the qualifier, the *Version* column displays the bundle version alongwith the timestamp.

2.6 Viewing Error Logs

Context

You use this procedure to view the errors of an integration content artifact for monitoring purposes.

Procedure

1. Launch SAP HANA Cloud Integration.
2. Choose *Window* > *Open Perspective* > *Other.*
3. In the *Open Perspective* dialog box, choose *Integration Designer*.
4. Choose *OK*.
5. In the *Node Explorer* pane, expand *cluster* > *<node name>* > *TM:<VM Name>*.

6. Choose the *Component Status View* tab.
7. In the context menu of an artifact in error state, select *Show Error Details*.

2.7 Tenant and Integration Flow Tracing

Tracing helps to track the message flow of processed messages and view relevant message payload at different points of message flow. It also helps to know if there is any error in the message execution.

To retrieve and view trace the following roles are needed.

Table 19:

Operations	Roles
View Message Trace	IntegrationOperationServer.read
View Message Payload/Header	esbmessagestorage.read
Export MPL	IntegrationOperationServer.read esbmessagestorage.read NodeManager.read

To enable tracing for integration flow, user needs to switch on tracing at tenant and integration flow levels. Tracing will work only for newly deployed integration flows from runtime version 1.11.

1. The 'Tenant Configuration' tab helps to enable tracing at tenant level. The following two properties enable tracing on corresponding worker node(s) for the specific tenant.
   a. String WN_IFL_TRACING_ENABLE = "wn.ifl.tracing.enable"- when set to true, enables tracing for worker node of type IFL. The default value is false.
   b. String WN_IFLMAP_TRACING_ENABLE = "wn.iflmap.tracing.enable"- when set to true, enables tracing for worker node(s) of type IFLMAP. The default value is false.
   c. For more details on configuring tenant configuration, please check operations guide.
2. 'Trace Configuration' enables tracing at integration flow level also.

Procedure

1. In the Model Configuration editor page, select the graphical area outside the integration flow pool.
2. In the Properties view, choose the Trace Configuration tab.
3. User can select trace level 'none',' body & header', 'header' from the dropdown menu of trace log, appearing in property sheet.

> **Note**
>
> Post message processing, the trace data is deleted after 24 hours.

View Message Processing Log

In the Message Monitoring editor, for a completed or failed message, the Properties view displays the message processing log (MPL) for this message. The MPL provides information on the steps during the processing of a

particular message and the View MPL button opens the particular integration flow with the message trace, displaying path of message flow. If the particular project does not exist in workspace then the project is automatically imported.

If the current project state is found to be different from deployed state then tracing of message flow occurs till the point, sequence of elements match.

On clicking the message icon in the message trace, payload and header tabs appear in property sheet. In case splitter is used, there are multiple message payloads and the icon has multiple message symbols. In such a scenario, the message heading is displayed between dashed lines and below the heading, payload content appears as shown below.

```
------------------------
Message 1
------------------------
<ns0:BSNMessage xmlns:ns0="https://bsnschemas.netweaver.neo.com/bsnflow">
<SenderId>Token</SenderId>
<ReceiverId></ReceiverId>
<MessageType></MessageType>
<FileName></FileName>
<NumberOfRecords></NumberOfRecords>
<MessageId></MessageId>
<MessageContent></MessageContent>
</ns0:BSNMessage>
```

One path can be traversed by multiple sequences. This information is displayed in the property sheet. For example, “Displaying content for message 1 (out of 3)”. Here, ‘3’ denotes number of times the particular path has been traversed. Clicking on the text enables user to switch to a different sequence.

In case of huge payload, the whole information will not be visible. In such a scenario, we can use ‘Export Payload’ button to view content using external tool/application.

Export Message Processing Log

Selecting a processed message in Message Monitoring editor, enables the Export MPL button. On clicking the same, Message Processing Log dialog box opens. User can select one of the following from the dialog box:

- Export Log – Message flow sequence only
- Export Log with Trace – Message flow sequence, message payload and headers at different trace points
- Export Log with Trace and Integration Project

> **Note**
>
> - View MPL and Export MPL works for one message only. Hence, even if more than one message is selected, View MPL and Export MPL will be valid for the first selected message.
> - You need all the roles mentioned in the table above, for any option you select from the dialog box.

Import Trace

To import the content, right click on the project explorer; select ‘Import’, select Message Processing Log Archive and click on ‘Next’. In the dialog box, select the archived file where the content is saved. The Integration Flow opens, decorated with the message trace.

> **Note**
>
> - Only content with trace can be imported.

- Tracing is only supported for messages which are in completed or failed state in Message Processing Log. For messages in retry state, tracing is not supported.

2.8 References to Additional Help

Cheatsheet

Cheatsheets guide you in performing simple end-to-end use cases. Cheatsheets are very useful if you are working with the tool for the first time and you want to understand the related steps to complete a task. It shows you the flow of steps and has UI controls to automatically open the relevant UI elements, such as a wizard or a view.

To open cheatsheets, follow the steps below:

1. From the main menu, choose *Help* > *Cheatsheets...* .
2. In the *Cheatsheet Selection* dialog, expand *SAP HANA Cloud Integration* folder.
3. Select the most appropriate cheatsheet from the list and choose *OK*.
4. Follow the instructions mentioned in the cheatsheet.

Context-sensitive Help

Context -sensitive help provides more details about a specific view, wizard or page. It is useful when you want more information about the parameters in the UI elements.

Select the UI element and press F1.

2.9 Understanding the Integration Content Types

SAP HANA Cloud Integration provides features on Eclipse Classic 4.2.2 to develop and configure integration content.

The feature, called the *Integration Designer*, provides options to develop integration flows in your local Eclipse workspace, which implies no network connection is required during development. Each integration flow is associated with a project and can refer to other entities, such as message mappings, operation mappings, and WSDL definitions, that are available within the same project.

The integration flow can also refer to an entity, such as a value mapping, that is not available within its project. You create a separate value mapping project such that the reference takes place across the projects within the workspace.

The integration flow along with other referenced entities form the integration content. Once you complete the development of integration content, you deploy the integration flow project as well as the referenced value mapping to the runtime.

> **Note**
>
> Another feature, called the *Integration Operation Monitoring*, provides options to monitor the deployed integration projects in runtime. For more information, see the Operations Guide for SAP HANA Cloud Integration.

Types of Integration Projects

The sections below introduce you to different project types that the tooling provides based on the entities.

Integration Flow Project Type

The *Integration Flow* project type contains packages for creating integration content, where each package consists of a particular entity.

Integration Flow Project Structure

Elements in Project Structure	Description
src.main.resources.mapping	Package for mappings to be used in scenario
src.main.resources.scenarioflows.integrationflow	Package for BPMN integration flow
src.main.resources.wsdl	Package for interfaces like IDOC , WSDL used in scenario
MANIFEST.MF	File contains dependencies to runtime components and integration content metadata

> **Note**
>
> Additional files that are available in the *Integration Flow* project types are:

Additional Elements in Project Structure	Description
src.main.resources	Package for parameters.prop and parameters.propdef files
parameters.prop	File contains externalized attributes representing a variable such as a the customer's landscape information.
parameters.propdef	File contains metadata such as value type of the parameterized attribute, its description and whether the attribute should be configured mandatorily.

Value Mapping Project Type

The *Value Mapping* project type is used for scenarios that require you to map different representations of an object to each other. Each value mapping project contains one or more value mapping group that is a set of values of the object.

Value Mapping Project Structure

Elements in Project Structure	Description
MANIFEST.MF	File contains dependencies to runtime components
value_mapping.XML	File contains value mapping groups that hold the objects and their corresponding representations

3 Packaging Integration Content in SAP HCI Spaces

You use this procedure to assemble integration contents into packages and publish them, so that integration developers can use these integration packages in their integration scenarios.

As an integration developer, you can now create integration packages for your specific domain or organization.

3.1 Importing Integration Packages

Context

Procedure

1. Choose *Import* in the *Design* tab to import an integration package.
2. Browse and select the integration package for importing.
3. Choose *OK*.
   Imported file appears in the *Design* tab of the application.

3.2 Creating an Integration Package

Procedure

1. Launch SAP HCI Spaces by accessing the URL provided by SAP.
2. If you are a new user, choose *Signup*.
3. If you have user credentials, choose *Login*.
4. Choose the *Design* tab.
5. Choose *Create*.
6. In the *<integration package name>* editor page, enter the required data.
7. Perform the following substeps as required:
   - If you want to add an integration flow as an artifact, perform the following substeps:
     1. Choose *Add* > *Process Integration* in the  (Artifacts) section.

2. Enter the required details in the *Create Process Integration Artifact* Artifact dialog box.
3. Choose *Browse* in the *Integration Flow* field to import an integration flow from your local system.
4. Choose the required file in the *Open* dialog box.
5. Choose *Open*.
6. Choose *OK*.

> **Note**
>
> Create a zip file of the required integration flow in your project folder before importing it as an artifact. The zip file must not contain any bin folder and you must not keep the integration flow in any sub folder.

- If you want to add a file as an artifact, perform the following susbsteps:

1. Choose *Add* > *File* in the  (Artifacts) section.
2. Enter the required details in the *Create File Artifact* dialog box.
3. Choose *Browse* in the *File Upload* field to import a file form your local sysytem.
4. Choose the required file in the *Open* dialog box.
5. Choose *OK*.

- If you want to add a mapping as an artifact, perform the following substeps:

1. Choose *Add* > *Value Mapping* in the  (Artifacts) section.
2. Enter the required details in the *Create Value Mapping Artifact* Artifact dialog box.
3. Choose *Browse* in the *File Upload* field to import a file from your local system.
4. Choose the required file in the *Open* dialog box.
5. Choose *Open*.
6. Choose *OK*.

> **Note**
>
> Create a zip file of the required value mapping in your project folder before importing it as an artifact.

- If you want to add an URL as an artifact, perform the following substeps:

1. Choose *Add* > *URL* in the  (Artifacts) section.
2. Enter the required details in the *Create URL Artifact* dialog box.
3. Choose *OK*.

- If you want to add a data integration as an artifact, perform the following substeps:

1. Choose *Add* > *Data Integration* in the  (Artifacts) section.
2. Enter the required details in the *Create Data Integration Artifact* dialog box.
3. Choose *OK*.

8. If you want to remove the integration package, choose *Delete Package*.
9. If you want keep the integration package, choose *Save*.
10. If you want to terminate the creation of the integration package, choose *Cancel* before saving it.

3.3 Working with an Integration Package

You use this procedure to perform various miscellaneous functions with the artifacts of an integration package.

Procedure

1. Launch SAP HCI Spaces by accessing the URL provided by SAP.
2. Choose the *Design* tab to view the list of integration packages.
3. If you want to view the details of an integration package, choose *<integration package name>*.
4. Choose ▶ ⚙ › *View metadata* to view the metadata of the artifact.
5. Choose ▶ ⚙ › *Deploy* to deploy the artifact.

   You can only deploy data flows, integration flows and value mappings. You can deploy multiple value mappings at a time.
6. Choose ▶ ⚙ › *Configure* to configure the artifact.

   You can configure only integration flows.
7. Choose ▶ ⚙ › *Download* to download an artifact.

   You can only download integration flow, files, and value mappings. Your local file system stores the downloaded artifact(s) with name as <IntegrationPackage>_artifact.zip file. If you want to use the artifact in eclipse, you need to import it in eclipse after extracting it from the zip file.

3.4 Editing an Integration Package

Procedure

1. Launch SAP HCI Spaces by accessing the URL provided by SAP.
2. If you are a new user, choose *Signup*.
3. If you have user credentials, choose *Login*.
4. Choose the *Design* tab.
5. Select the required integration package.
6. In the *<integration package name>* editor, choose *Package Content*.

7. Choose *Edit*.
8. If you want to edit integration content of type process integration, data integration, file or value mapping, choose 

   You get the following functions:

   - View metadata
   - Delete
   - Download
   - Configure (only for integration flows)
   - Deploy (only for data flows and integration flows)

9. If you want to edit artifact of type URL, choose  .

   You get the following functions:

   - View metadata
   - Delete

10. If you want to edit the metadata of an artifact, then perform the following substeps:
    a. Choose *View metadata*.
    b. In the *<Artifact Name Details>* dialog box, choose *Edit*.
    c. Choose *Save* to keep the changes.
    d. Choose *Save as version* to retain a copy of the current artifact.

       You can view the version history of an artifact by choosing the current version mentioned along with it.
    e. Choose *Cancel* to revert the changes.
11. If you want to edit a specific integration package that is present in the *Discover* tab, then perform the following substeps:
    a. Choose the *Discover* tab.
    b. Select the integration package.
    c. Choose  to copy it to the *Design* tab.

    You can copy an integration package to the *Design* tab by opening the integration package in the *Discover* tab and choosing *Copy* in the integration package editor. If the copy fails, then choose Create copy to create a copy of the integration package or Overwrite to erase errors and register new information in the integration package.
12. If you want to download multiple artifacts at once, then perfomr the following substeps:
    a. Select the *Artifacts* checkbox.
    b. From the list of artifacts, select the ones you want to download.
    c. Choose *Actions* > *Download all*
13. If you want to keep the changes made to the integration package, choose *Save*.
14. If you want to discard the changes, choose *Cancel* before saving it.

3.5 Locking Integration Packages

You use this procedure to restrict multiple users from editing an integration package.

Context

Procedure

1. Launch SAP HCI Spaces by accessing the URL provided by SAP.
2. Choose the *Design* tab.
3. Choose an integration package from the list.
4. Choose the *Edit* button.

   When you choose *Edit* in an integration package editor, it locks the integration package and prevents any other user from modifying it.
5. Choose *Save* after making the required changes.
   - Choosing *Save*, *Cancel* or *Delete* releases the integration package and allows it to be modified by other users.
   - In case you close your browser without saving the integration package or there is a session timeout for you, the integration package remains locked by you unless you edit and save, cancel or delete it.

3.6 Exporting Integration Packages

Context

Procedure

1. Choose the *Design* tab.
2. Select the required integration package from the list.
3. 3. Choose *Export*.

The destination of export depends on the default browser setting. By default, the downloaded file takes the name of the integration package. In case the name contains special characters, the browser changes the file name by appending them.

4 Accessing Integration Content in SAP HCI Spaces

SAP HCI Spaces is a Web-based application that helps you to access integration content available for a particular tenant on an OnDemand integration infrastructure.

Types of integration content that you can access in SAP Spaces Application are:

- Integration flows and their associated configuration details
- Integration packages with artifacts such as, value mappings, integration flows, and files

Using SAP HCI Spaces you can:

- Visualize the list of integration flows deployed on a tenant
- View and edit the configurations associated with the deployed integration flow
- View configuration details of the integration flow elements
- View the list of integration packages
- Download the required artificats from the integration packages

4.1 Viewing Integration Flow Configurations

Context

SAP HCI Spaces allows you to view the deployed integration flows and their associated configurations.

Procedure

1. Launch the *SAP HCI Spaces* application by accessing the URL provided by SAP.

   > **Note**
   >
   > Browsers that support the application are *Internet Explorer 9*, *Google Chrome* and *Safari*.

2. To view the list of integration flows, choose  and select the *Discover* tab page.
3. If you want to filter and sort the integration packages, perform the following sub-steps:

   a. If you want to apply filter condition, choose  .
   Apply Filters screen area is activated.
   b. Select the product and keyword you want to filter for and choose *Ok*.
   c. If you want to sort according to name or date, choose the appropriate option from the drpdown list

4. To view details of an integration flow, select the bundle name of the integration flow in the tabular view and the integration flow you want to view.
5. To see the configuration details of an element in the integration flow diagram, select it.
You can view the configuration details of the selected element below the diagram.

> **Note**
>
> To view a specific configuration detail about the element, select the required tab in the configuration view.

4.2 Configuring Multiple Integration Flows

Prerequisites

You have copied the integration package to your workspace

Context

SAP standard integration packages offer a feature where you can configure multiple integration flows that connect to the same system at once. You need to enter the configuration details only once in the mass configuration screen. All the integration flows that connect to that system instance are updated with the details. This eliminates the need for individually entering the configuration details in each integration flow.

For example, consider a scenario in which all the integration flows connect to the same instance of SAP ERP. The configuration details like Host, Port and Client are the same for all the integration flows. Once you specify these details in the mass configuration screen, the application updates these details in all the relevant integration flows. This simplifies the task of configuring multiple integration flows that share common system properties.

> **Remember**
>
- The application consolidates the field names into a single section only if they match exactly across all the integration flows you select for configuration.
- Mass configuration is supported for integration flows with SOAP or iDoc adapters only.
- You can also configure authentication details during mass configuration.

Procedure

You use this procedure to configure multiple integration flows.

1. Launch SAP HCI Spaces using the application URL.
2. Choose ☰ > *Design* tab page.
3. Select the package that contains the integration flows that you want to configure.
4. In the *Artifacts* screen area, choose the checkbox relevant to the integration flows you want to configure.
5. Choose *Actions* > *Configure*.
   The application displays the externalized system properties pertaining to the selected integration flows that you want to configure. If there are common properties, the application displays them as a single entry.
6. Enter the system properties and choose *OK*.
7. If you want to save the configuration, choose *Save*.
8. If you want to deploy the integration flows, choose *Deploy*.

4.3 Editing Integration Flow Configurations

Context

You use this procedure to edit endpoint and certificate configurations for a deployed integration flow. The application displays configuration details only for externalized parameters.

> **Note**
>
> To externalize parameters for a deployed integration flow, access the Eclipse environment of SAP HANA Cloud Integration.

Procedure

1. To edit the endpoint and certificate configurations of the required integration flow bundle, go to ☰ > *Design* page and choose the package that contains your integration flow you want to edit.
2. Select the required integration flow and choose *edit*.
   The fields that can be edited are activated and you can enter or edit the details. When you select an element in the integration flow, the system displays editable details for that particular element and you can provide the same.
3. To deploy the changes on the current tenant, choose *Deploy*.
4. To view the changes, choose the icon *(Refresh)* on the *Run* page.

4.3.1 Configuring SuccessFactors Adapter

Context

You use this procedure to configure the system and channel properties for SuccessFactors adapter.

Procedure

1. Launch SAP HCI Spaces by using the application URL.
2. Choose the *Design* tab.
3. Select the required integration bundle and integration flow from the list.
4. Choose the SucessFactors adapter connection in the editor screen.
5. Choose the *Adapter Specific* tab in the *Channel* section.
6. Choose *Edit*.
7. Choose *Model Operation*.
8. If you want to add a new system, then in the *System* dialog box that appears, choose *Add System*.
9. Enter the following details:
    - System
    - Address
    - Company Id
    - Username
    - Password
10. Choose *Connect*.
11. Make required configurations in the *Query Builder* section.
12. Choose *OK* to save the changes.

4.3.2 Configuring SFTP Adapter

Context

Procedure

You use this procedure to configure the SFTP adapter.

1. Launch SAP HCI Spaces using the application URL.
2. Choose the *Design* tab.
3. Select the required integration package from the list.
4. Select the required integration flow from the list and choose *Edit*.
5. In the integration flow, select the communication channel containing the SFTP adapter.
6. In the *Channel* screen area, select the *Adapter Specific* tab page.
7. Provide values in fields based on description given in table below.

Field	Description
Directory	Directory in the SFTP server that you are accessing
File Name	Provide the name of the file
Append Timestamp	Select if you want the timestamp to be added to the file
Server Host	URL of the host server that you are connecting to
User Name	User name that you are using for authentication

8. To save the configuration, choose *Save*.

4.3.3 Configuring OData Adapter

Context

The OData adapter enables you to communicate with OData service providers. You can use this adapter in integration flows that involve systems that support OData service.

Procedure

You use this procedure to configure the OData adapter.

1. Launch SAP HCI Spaces using the application URL.
2. Choose the *Design* tab.
3. Select the required integration package from the list.
4. Select the required integration flow from the list and choose *Edit*.
5. In the integration flow, select the communication channel containing the OData adapter.
6. Choose the *Adapter Specific* tab page.
7. If you wish to modify the operation, manually enter the operation that you want to perform in the *Operation* section.

You can only enter `Query (GET)` or `Read (GET)`

8. To modify the operation details further, perform the following substeps:
   a. Choose *Modify Query*.
   b. If you have already configured a system, select it from the dropdown list and choose *Connect*.
   c. If you have not configured a system, choose  to add a system.
   d. Provide values in fields based on the description given in table below and choose *Connect*.

Field	Description
System	Name of the system that you are adding **Note** You can use this name as a reference when you configure the OData adapter again.
Address	URL of the system that you are connecting to
Username	User name that you are using for authentication
Password	Relevant password for the user name

   e. To modify the filter conditions, choose  and make the required configurations.
   f. To modify the sorting conditions, choose  and make the required configurations.

> **Note**
>
> Sorting is not available for `READ` operation.

   g. Choose *OK* to update the operation with the changes.
9. If you want to save the operation, choose *Save*.

4.3.4 Scheduling SFSF Adapter for Querying Data from SuccessFactors System

Context

Procedure

1. 1. Launch SAP HCI Spaces by using the application URL.
2. 2. Choose the  *Design* tab.
3. 3. Select the required integration flow from the list.
4. 4. Choose the SucessFactors sender adapter connection in the editor screen.
5. 5. Choose the *Adapter Specific* tab in the *Channel* section.
6. Choose the *Scheduler* tab.
7. Choose *Edit*.
8. If you want to schedule the adapter on specific day(s), then perform the following substeps:
    - Select *On Date* and enter the required date.
    - Select *Daily* for daily updates.
    - Select *Weekly* and choose the required weekday(s).
    - Select *Monthly* for monthly updates.
9. If you want to schedule the adapter on a time basis, then perform the required substeps:
    - Select *Time* and enter the required time.
    - Select *Every ___ Minutes Between ___ HH and ___ HH* to schedule a minute based update.
    - Select *Every ___ Hour Between ___ HH and ___ HH* to schedule an hourly update.
10. Select the time zone in the *Time Zone* field.
11. Choose *Deploy*.

4.3.5 Configuring PGPEncryptor and PGPDecryptor

Context

Procedure

You use this procedure to configure the parameters of the PGPEncryptor and PGPDecryptor.

1. Launch SAP HCI Spaces using the application URL.
2. Choose the *Design* tab page to access your workspace.
3. Select the required integration package.
4. Select the required integration flow and choose *Edit*.
5. Choose the *PGPEncryptor* or *PGPDecryptor* in the integration flow.
6. In the integration flow editor, provide the details based on the field descriptions given in the table below.

Table 20: Fields and description for PGP Encryptor

Field	DFields and descriptions for PGP Encryptore-scription
Name	Name of the PGP Encryptor element
Signatures in PGP Message	Select whether or not to include a signature in the PGP message
Content Encryption Algorithm	Select the encryption algorithm that the PGP Encryptor uses
Secret Key Length	Select the length of the secret key to be used for encryption
Armored	
With Integrity Packet	
Encryption User ID of Key(s) from Public Keyring	Enter the User ID of the encryption key that you are using and choose +
Digest Algorithm	Select the digest algorithm to be used for the signature
Signer User ID of Key(s) from the Secret Keyring	Enter the User ID of the signer key that you are using and choose +

Table 21: Fields and descriptions for PGP Decryptor

Field	Description
Name	Name of the PGP Decryptor element
Signatures in PGP Messages	Select whether or not to include a signature in the PGP message
Signer User ID(s)	Enter the User ID of the signer key that you are using and choose +

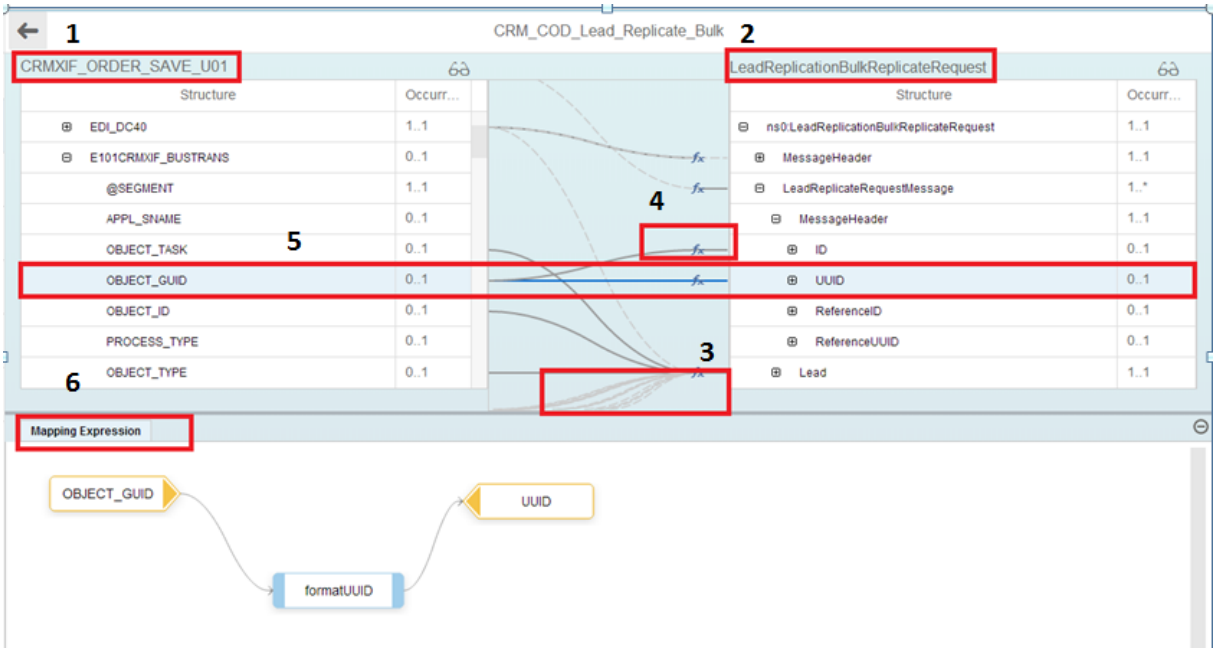
7. If you want to save the configuration, choose *Save*.
8. If you want to deploy the integration flow, choose *Deploy*.

4.4 Viewing Mapping Details in Mapping Viewer

Context

You use this procedure to view mapping details of mappings appearing in an integration flow.

To familiarize yourself with the various features and behavior of elements in the mapping viewer, see the screenshot below:



Sections marked in above screenshot	Description
1	Represents the source structure.
2	Represents the target structure.
3	If any of the entities (source or target) are not visible, a dotted line indicates the calculated position of the invisible entity.
4	Solid lines represent visible source and target entities.
5	If you select a line, both source and target entities come to the center in the table, and the line becomes solid.
6	This section represents the functions of the selected mapping.

Procedure

1. If you are viewing an integration flow from the *Catalog* view, follow the sub-steps below:
   a. Select an integration package.
   b. In the *Artifacts* table, select the integration flow name from the *Name* column.

   > **Note**
   >
   > In the *Artifact* table, the integration flow *Type* appears as *Process Integration*.

2. If you are viewing an integration flow from the *My Projects* view, select an integration flow from the *AVAILABLE INTEGRATION FLOWS* table
3. In the integration flow diagram, select the mapping element.

   Hover over the different entities in the diagram to identify mapping elements.

   The mapping name appears as a link in the parameter section below the diagram.
4. To view the mapping in the mapping viewer, select the link with the mapping name.

   > **Note**
   >
   > If a mapping is invalid or unsupported, the mapping viewer does not open the mapping. If you hover over the mapping name link you can view the associated error message.

5. In the mapping viewer, select a source or target entity to highlight all mappings from that entity.

   You can select message mappings between source and target entities to view the functions associated with them. The application displays the functions in the *Mapping Expression* section.

4.5 Editing Mapping Details

Prerequisites

You have opened the mapping in mapping viewer.

Context

You use this procedure to edit the mappings of an integration flow in SAP HCI Spaces application.

Procedure

1. Choose the *Design* tab.
2. Choose an integration package.
3. To edit a mapping in mapping viewer, choose the required integration flow.
4. Choose *Edit*.
5. To add a mapping , connect source elements to target elements using connectors.

   > **Note**
   >
   > Only direct mappings, that is, 1:1 mapping can be created

6. To edit a mapping graphically, perform the following substeps:
   a. Choose the required integration package.
   b. Choose *Edit*.
   c. Select the required mapping in the integration flow editor.
   d. In the *Mapping* section, choose *<mapping name>*.
   e. In the *<mapping name>* editor, choose the required mapping expression.
   f. In the *Mapping Expression* section, edit the mapping accordingly by providing the required functions and values.
7. To delete any existing mapping, select the mapping and delete.

   > **Note**
   >
   > - You can also select *Delete* from the context menu of the mapping connector.
   > - You cannot perform a partial deletions in mapping viewer, you can only delete the entire mapping.
   > - If you delete a mapping with functions, all the associated functions are also lost and cannot be recreated in Web.

8. Choose *Ok*.

   This action triggers a validation check, which reports an error if there are any unassigned mandatory fields.
9. To revert the changes, choose *Cancel*.

   If you have performed series of changes to the mapping then all your changes will be lost .
10. In the integration flow viewer, select *Deploy*.

    If you select *Cancel*, all the changes to the mapping will be lost.

4.6 Changing Source and Target Message Structuring

You use this procedure to extend the source or target or both message structures of a message mapping to the original message for containing extended fields.

Context

Procedure

1. Launch SAP HCI Spaces by accessing the URL provided by SAP.
2. Choose the *Design* tab.
3. Select the required integration flow with mapping.
4. Choose the mapping to view it in *Mapping Viewer*.
5. Choose (Edit) to open the mapping in *Mapping Editor*.
6. In the *Source* and *Target Messages* view, choose (Edit) to modify the required source or target message.
7. Choose the required file in the dialog box that appears.
   The file gets replaced with the chosen file.
8. Choose *Open*.

4.7 Working with Value Mappings

You use this procedure to perform various functions in a value mapping.

Context

Procedure

1. Launch SAP HCI Spaces by accessing the URL provided by SAP.
2. Choose the *Design* tab to view the list of integration packages.

3. Choose an integration package.
4. From the list of artifacts, choose the required value mapping.
5. Choose *Edit*.
6. In the *Bi-directional mappings for* section, choose *Add* to add a bi-directional mapping.
7. In the *Value mappings for* section, choose *Add* to add a value mapping.
8. Choose  to remove a bi-directional or value mapping.

   You can choose *Delete All* to remove all bi-directional and value mappings from the respective secions.
9. Choose *Save* to keep the changes.
10. Choose *Save as version* to retain a copy of the current artifact.

    You can view the version history of an artifact by choosing the current version mentioned along with it.
11. Choose *Cancel* to revert the changes.

> **Note**
>
> If you edit a web edited value mapping in eclipse, then you get a default value for 1:N, M:,1 and M:N mappings. SAP recommends that all groups should contain only one agency identifier and value pair.
>
> Some of the function names in web UI differ from the ones in Eclipse.

4.8 Deploying Data Flows

Context

Procedure

1. In the *Integration Package Editor*, in the *Artifacts* section, select the required data flow from the list.
2. Choose  *Deploy*.
3. In the *Settings* dialog box that appears, enter the following details:
   - Data Center URI
   - Application Name
   - Organization Name

   Once you enter the above mentioned details, the application forms the Data Service URL.
4. Choose *OK*.
5. In the *LOG ON* screen that appears, enter the organization name.
6. Choose *Log On*.
   The Data Integration application opens its *Projects* tab.

7. In the dialog box that appears, enter the following details as required:
    - Name
    - Source
    - Target
8. Choose *OK*.
   The newly deployed data flow appears as an entry in the *Projects* tab page. If required, you can switch back to the SAP HCI Spaces application.

> **Note**
>
> To switch to SAP HCI Spaces Catalog view from Data Integration using the `Back Button` in the browser, perform the required substeps:
>
> - If you use Internet Explorer, from the context menu of `Back Button`, select HCI.
> - If you use Google Chrome, click twice on `Back Button`.
> - If you use Mozilla Firefox, from the context menu of `Back Button`, select HCI.
>
> If you want to switch to SAP HCI Spaces *Catalog* view from Data Integration using the `Forward Button` in the browser then by default, it displays the *Projects* tab page without the Settings option. This behavior remains the same for Internet Explorer, Google Chrome, and Mozilla Firefox.
>
> If you do not have authorization to access the Data Integration application then you get the following error messages:
>
> - Error message for Google Chrome and Mozilla Firefox:
>   User is not included in the organization. Contact your security administrator for assistance.
> - Error message for Internet Explorer:
>   HTTP 404: Not found error

Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

Accessibility

The information contained in the SAP documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of wilful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: http://help.sap.com/disclaimer).

www.sap.com/contactsap